RX11

RX11 INTFC
CZRXBF0

AH 9341F MC

FICHE 1 OF 1

NOV 1979
digital
MADE IN USA

PRODUCT CODE: AC-9339F-MC

PRODUCT NAME: CZRXBF0 RX11 INTERFACE DIAGNOSTIC

DATE: APRIL 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: DAVID L. ADAMS
MODIFIED BY MIKE PAGE TO REV F

CONTENTS

1.0 GENERAL PROGRAM INFORMATION

1.1 Abstract

The RX11 interface diagnostic consists of a series of selectable tests that may be run individually, sequence through all tests, or start at a selected test and run through remaining tests, in order, then go back to the selected test.

These tests check out the basic functions of the RX11 interface such as:

1. Done flag
2. Interrupt level / address
3. Program initialize
4. Read status registers
5. Fill / empty buffer transfer verification
6. Fill / empty buffer with data patterns

It is necessary to insure that these functions work before a data reliability test is run.

Any errors are reported by the program, and it is possible to loop on the error or a particular test for scope testing.

1.2 System Requirements

1.2.1 Hardware Requirements

The following equipment is required:

1. PDP-11 series computer with minimum of 8k memory
2. RX11 floppy disk system, including a single or dual drive RX01 and a PDP-11 Interface card [m7846].

NOTE

A diskette must be included with each drive tested.

3. Console teleprinter

1.2.2 Software Requirements

No prerequisite software

2.0 OPERATING INSTRUCTIONS

2.0.1 Outline of Operating Procedure

The standard running procedure for the diagnostic (to run all tests on both drives with no operator intervention via the switch register) is as follows:

1. Load the program into memory

    1. If it is being loaded from a diskette replace the "library" diskette with a "scratch" diskette.

NOTE

If this step is forgotten and the program was loaded via RXDP (floppy monitor) on unit 0 with unit 0 selected by user to undergo testing the program will failsafe the operation and prompt the user as follows: "Caution - If you desire to test unit 0 replace load medium with a scratch diskette then press continue"

Caution again, however ----------

NOTE

When running this program on a Small 11 (e.g /04, LSI 11, etc.) Where there is no console switch register it is imperative to remember this setup.

NOTE

Before proceeding to Step B, ensure that the following modifiable locations contain the parameters you require for testing. The following table describes each location with respect to the default parameters which will be used if left unmodified by the user:

LOCATION	LABEL	CONTENTS	PROGRAM REACTION
1200	OD:	0	TRACKS 0,52,53,114(8)
1202	FIRST:	015001	SECTORS 1 THRU 32(8)
1204	KRXVEC:	264	ASSUMES PROPER DEVICE VECTOR
1206	RXCS:	177170	ASSUMES PROPER DEVICE STATUS REGISTER (CALCULATES 'RXDB' ADDRESS FROM)
1212	DTESTP:	0	TESTS BOTH UNITS AUTOMATICALLY SEQUENCES THRU ALL TESTS
1214	BRLEV:	5	ASSUMES PROPER DEVICE 'BR' LEVEL

Reference section 2 of this document for a more thorough description of each of these items and how to modify these locations if you desire to change the above mentioned default testing parameters.

2. Start the program at location 200

3. The program will type out maindec number, a test parameter of 0 (use both drives and run all tests). Then type tracks to be accessed and sector limits. The program is now running all tests in sequence.

4. If there are no errors, at the end of the pass (approx. 50 seconds run time), a "D" will be typed and it will continue on for another pass.

5. To halt the test at any time (after or before completion of a pass) just halt the processor.

6. After completing a pass of the diagnostic, the RX11 reliability test may be run.

7. There are two types of error print out formats:

   1. Tests pretest, 1 - 17, and 21 - 23 use the format shown in section 3.1. The important address there is the "ERADR" (Error Address) go to the listing at that

location to get more information on the error condition

2. Test 20, and 24 - 26 use the format shown in section 3.3. In this case the "TEST PC" is the address of the test being run when the error occurred. Then the vital information of the error is printed (contents of all registers, address of where on the diskette the error occurred, and the type of error).

2.1 Loading The Program

Load the program into memory using the standard procedure for binary paper tapes. Make sure the total system is ready for operation. The diskettes inserted properly, doors closed on drives to be tested etc.

2.2 Starting Addresses

The program has two starting address locations as follows:

2.2.1 Initial Start [Loc.200]

This starting address tests for and selects the hardware, or software switch register, prints maindec name and revision, the test and drive selection, and tracks and sectors being used.

2.2.2 Restart [Loc.202]

This starting address directs the program to continue running using the drive and test selections specified in the previous initial start.

2.3 Operator Action Before Starting The Program

2.3.1 Device Address Selection

Like most options on the PDP-11 the RX11 Interface Card has jumperable register and vector addresses. This allows for devices with the same standard addresses to be jumpered to an other address so they will run without conflict.

The program must know what addresses are being used, as it is through these register and vector addresses that all communication between the

PDP-11 and the RX11 is handled.

If the RX11 system under test is jumpered for register addresses other than standard, which is RXCS = 177170 and RXDB = 177172 place in the memory location called "RXCS" [Loc.1206] its new address. The program assumes the next even address above that of RXCS, will be the address of RXDB, so setting that address is not necessary. If there is a nonstandard interrupt vector address (standard is Loc. 264) then place in memory location called "KRXVEC" [Loc. 1204] its new address.

If either of these locations is loaded with a wrong address, the program will get unpredictable errors and may halt.

NOTE

The program expects that the priority level jumpers are set for a normal 'BR' level of 5 (contents of program location 'BRLEV:' is set to 5). If the priority level jumpers are set to any other level tests 3 & 4 will report errors, unless program location 'BRLEV:' has been patched to contain the relevent 'BR' level before executing the program.

If this is being tested on a LSI 11, tests 3 and 4 will not be run as the LSI 11 has only 1 level of interrupt.

2.3.2 Non-Standard Diskette Address Selection

If it is desirable to test the diskette between track and sector address limits other than the preselected track addresses, and/or minimum (first) and maximum (last) sector addresses, this is done by the operator making changes to two memory locations before the program is started. One location is called "OD" [Loc. 1200] which contains the two bytes for inner and outer track addresses. The other location is called "FIRST" and it contains the two bytes for the first and last sector addresses.

1. Definitions
   OD = Address of track at outer diameter (min. 0)
   ID = Address of track at inner diameter (max. 114)
   FIRST = Address of first sector on a track (min. 1)
   LAST = Address of last sector on a track (max. 32)

2. Locations

Tracks location 1200	bits	14----8	6----0
		ID	OD
Sectors location 1202	bits	12----8	4----0

SEQ 0009

LAST FIRST

3. Restrictions

The value of "OD" must be less than or equal to the value of "ID". The value of "FIRST" must be less than or equal to the value of "LAST".

If these locations are changed to new limits, then the program will access only those addresses inclusive of and between these limits. The exception to this is test 26 which always uses a special track sequence.

If the "OD" location is cleared or set to any illegal combination of tracks, the program will clear location "OD". The track sequence will then be tracks 0, 52, 53, and 114 (octal) only.

If the "FIRST" location is cleared or set to any illegal conbination of sector addresse limits then the program will set "FIRST" to 1 and "LAST" to 32 (octal).

2.3.3 Software Switch Register (Loc. 176)

For the PDP 11 processors that do not have a hardware switch register or if the operator wishes to select the software switch register, by putting all the switches up to a "1", (this must be done each time the program is started at location 200, otherwise the program will use the hardware SWR.) Location 176 is assigned as the switch register. Bits set to a "1" in this location have the same function as the corresponding switch in the hardware switch register. All references to the SWR are indirect and the program assigns the correct address of the SWR at "initial start". See Section 2.4.2 for the selection of operating conditions.

To change the software SWR. while the program is running, type "control G". Each time the SWR. is to be tested the program will check to see if the software SWR is selected, and the program is not running in auto mode of RXDP/ACT11. If both conditions exist then the program checks for the CTRL G in the keyboard buffer. If the CTRL G is there the contents of the software SWR. are printed and a "new =" is asked for. The operator may now type in the new switch register contents, terminated by a carriage return (CR), or if he doesn't want to change the SWR. just terminate with the (CR). Note see the character restrictions below.

When the program detects the (CR) it will replace the contents of the software SWR., if a new one has been typed in, and return to the flow of the program.

NOTE

Character restrictions for changing the software SWR.

1. Only octal numbers 0 - 7 are accepted. Any other character typed will be printed as a ? and the whole SWR must be retyped.

2. To wipe out a "new" contents just typed in, type CTRL U. Now a new contents can be retyped.

3. Only 6 octal characters will be put into the SWR. If more than 6 characters are typed in only the last 6 will be put into the SWR.

2.3.4 Test Parameter Selection ("DTESTP" Loc. 1212)

The drive and test delection must be made before the program starts. Location "DTESTP" (Loc. 1212) is where the bits are set to tell the program what drives are wanted and what tests to run as indicated below. When the program starts it will print out the conditions under which it is running.

Bit 15 (1) Select Drive Unit 1
Bit 14 (1) Select Drive Unit 0

NOTE

If neither of the above bits are set to a 1, then the program expects both drives to be ready for operation (power on, diskettes inserted, and doors closed).

SEQ 0011

Then set the test selection in bits 4,3,2,1,and 0 as follows:

"DTESTP" BITS 15 14 13-----5 4 3 2 1 0
U1 U0 NOT USED TESTS

BITS 4	3	2	1	0	TESTS
0	0	0	0	0	(IF NO TEST SELECTED DEFAULTS TO TEST 1)
0	0	0	0	1	TEST 1
0	0	0	1	0	TEST 2
0	0	0	1	1	TEST 3
0	0	1	0	0	TES1 4
0	0	1	0	1	TEST 5
0	0	1	1	0	TEST 6
0	0	1	1	1	TEST 7
0	1	0	0	0	TEST 10
0	1	0	0	1	TEST 11
0	1	0	1	0	TEST 12
0	1	0	1	1	TEST 13
0	1	1	0	0	TEST 14
0	1	1	0	1	TEST 15
0	1	1	1	0	TEST 16
0	1	1	1	1	TEST 17
1	0	0	0	0	TEST 20
1	0	0	0	1	TEST 21
1	0	0	1	0	TEST 22
1	0	0	1	1	TEST 23
1	0	1	0	0	TEST 24
1	0	1	0	1	TEST 25
1	0	1	1	0	TEST 26

NOTE

Selection of tests 27 through 37 will cause the message "illegal test" to be printed.

NOTE

When a specified test is selected the program will start at that test and then run through all the following tests until it completes test 26, indicated by the EOP type out. Then it will go back to the test selected and start the next pass. (i.e. If test 24 is selected the program will rn test 24, 25, and 26, then go back to test 24.)

An expanded definition of the tests is in section 2.5

2.3.4.1 Prerequisite Of Tests: - The following tests must be run in order, as one test sets up for the next test.

Test 6 before tests 7 and test 10
Test 14 before test 15 and test 16
Test 16 before test 17
Test 21 before test 22 and test 23

See section 2.5 under the above tests for explanation

2.4 Operator Action To Run The Program

2.4.1 Starting the Program

Depending upon the starting address selected the program will do the following:

SA200 (Initial Start)

The selection of hardware or software switch register is made then the program will type its identification number, the test parameters selected in location "DTESTP", and tracks and sectors being tested. The program then proceeds to run under those conditions.

SA202 (Restart)

The program will type out the test parameters selected by the previous initial start, prints the diskette address limits, and starts running the tests. The only operator action required is to set the operating conditions as defined in Section 2.4.2, after depressing the "LOAD ADRS" switch and before depressing the start switch.

2.4.2 Operating Conditions

After tne test selection has been made press the "CONT" switch. The program will then ask for operating conditions. Switches 0 and 8 through 15 are used as indicated below. Once they are set up again depress the "CONT" switch. The program is now running under the selected conditions.

SW15-SW0 (1) - Select Software Switch Register

NOTE

If there is a hardware switch register, and the operator wants the software switch register, put all switches up (1) befort starting the program at the initial start address.

SW15 (1) - Halt on Error

The program halts on detecting an error, after printing the error message. Pressing "CONT" restores the normal operation of the program.

SW14 (1) Halt at End of Pass

At "end of pass" the program types a bell then an EOP indicator.

"D" means no errors during the pass
"-" means had errors during the pass

If SW14 is set the program will halt. If SW14 is off the program goes back to the test selected and recycles through to the last test, at which time another EOP indicator is printed. If the program halts due to SW14 then press "CONT" will restore the normal flow of the program. If it halts at the end of a pass it will type out the number of passes completed.

SW13 (1) - Inhibit Error Typeout

At the detection of an error if SW13 is set no error print out will occur. If SW13 is off the error information is printed as described in section 3.0 error detection.

SW12 (1) - Loop on Test

At the completion of a test the program checks SW12. If set the program will go back to the beginning of that test and rerun it. This produces a scope loop on a particular test. The program will stay in this test until:

1. Halt on end of test switch is set

2. Loop on test switch is turned off

At which time the program will go on to the next test.

NOTE

If SW12 is set and no test specified (0) the program will loop on test 1.

NOTE

To loop on a test that requires a previous test to be run first (Section 2.3.4), select the prerequisite test and set the "HALT AT END OF TEST" switch. Start the program and when it halts, select the desired test and set the "LOOP ON TEST" switch. The program will now stay in that test.

SW11 (1) - Lock on Error

In some tests errors can occur in several places throughout the test. When the error has been reported the program sets a PC flag to indicate where the error occurred. If SW11 is set the program goes back to the beginning of the test running, and goes through the test until:

1. It finds a different error in an earlier part of the test in which case it will lock onto that error.

2. It detects the PC flag indicating this is where the error occurred. It then goes back to the beginning of the test again.

This loop will continue until halt on error switch is set or the lock on error switch is turned off.

SW10 (1) - Halt at End of Test

When set it will halt the program at the end of the test presently running.

SW 9 - Limit Data Error Print Outs

(0) - When off only the first 10 data byte errors will be printed on a read check test, for each sector. Any more errors will be tabulated but not printed. An error on a different sector will allow 10 more data byte errors to be printed.

(1) - When set all data byte errors for all sectors will be printed on an error.

SW 8 (1) - Inhibit Recalibration

No recalibration of the drives will occur upon the detection of a seek error if this switch is set.

SW 0 (1) - Inhibit Bell at Error

If SW0 is off the error routine will ring the teleprinter bell at each error detected. With SW0 set no bell will ring.

2.5 Test Definitions

2.5.1 Pretest - Initialize [Key] Part I

Each time the program is started, by either starting address, it runs through a pretest.

Key initialize should set the done flag because any initialization of the RX01 microprocessor is an implied [read sector] of track 1 sector 1. Therefore any error, except parity, that may occur from a normal [read sector] command may occur during an initialize, causing the error flag to set.

Pretest insures that:

1. Done is set
2. Error flag is cleared
3. TR flag is cleared
4. Init done is set

2.5.2 Test 1 - RXCS Test Part I / Interrupt Test Part I

The purpose of this test is to verify that writing all RXCS writable bits to a 0 are not written to a 1.

The program writes the RXCS = 0

No interrupts should occur

The RXCS should remain unchanged = 40 (done)

The RXDB should = 0

2.5.3 Test 2 - Interrupt Test Part II / Vector Address Verification

The purpose of this test is to verify that writing the RXCS interrupt enable bit (bit 6) to a 1, does indeed write it to a 1, therefore because done is set an interrupt should occur (the PDP 11 priority is 0)

2.5.4 Test 3 - Interrupt Test Part III / Priority Level Test Part I

The purpose of this test is to verify the priority of the interrupt request line. The program sets the PDP-11 priority to 4

An RX01 interrupt should occur on priority level 5

If no interrupt occurs then the priority level of the RX11 is not 5, but maybe levels 4,3.2,or 1

2.5.5 Test 4 - Interrupt Test Part IV / Priority Test Part II

The purpose of this test is to verify the priority of the RX11 interrupt request line. The program sets the PDP-11 priority to 5.

No interrupt should occur. If an interrupt does occur then the priority level of the RX11 is not level 5, but maybe level 6, or 7.

2.5.6 Test 5 - Init [Programmed] B / Read Status

The purpose of this test is to verify that setting the RX11 bit 14 causes a RX01 programmed subsystem initialize.

The RXCS should = 40 (done)
The RXDB should = 4, or 104, or 204, or 304

Test 5 cont'd - RXCS test part II / RST

The purpose of this test is to verify the read status command (Function #12)., and that done bit is cleared by the function.

2.5.7 Test 6 - Fill Buffer Transfer Length Test

The purpose of this test is to verify the transfer length of the function "fill buffer" of the RX01 microcontroller

NOTE

This test loads the sector buffer for test 7 and 10, and must be run previous to them.

2.5.8 Test 7 - Empty Buffer Transfer Length and Content Verification Part I

The purpose of this test is to verify the transfer length of the function "empty buffer" and to verify the contents of the sector buffer.

2.5.9 Test 10 - Empty Buffer Transfer Length and Content Verification Part II

The purpose of this test is to verify the previous empty buffer test did not empty and destroy the contents of the sector buffer.

2.5.10 Test 11 - Fill / Empty Buffer With All 0"s

During the empty buffer function this test verifies that all 0's are in fact in the sector buffer.

2.5.11 Test 12 - Fill / Empty Buffer With All 1's

During the empty buffer function this test verifies that all 1's are in fact in the sector buffer.

2.5.12 Test 13 - Drive Ready Verification

Tests that the drive ready (RDY) bit will set for all selected drives. The RDY bit will be set after a read status function directed to the the selected drive.

2.5.13 Test 14 - Error Flag and B-Code Verification Part I

The purpose of this test is to verify that trying to read a non-existant sector will cause an error and the correct error code will be put into the RXDB when the status B is read.

NOTE

This test checks for parity error on the read status B function, the next two tests (T15 & T16) do not. This test must be run before tests 15 & 16.

2.5.14 Test 15 - Error Flag and B-Code Verification Part II

This test verifies that trying to write deleted data on an illegal sector will produce an error and the correct B-code is produced. The deleted data bit should be set after this test.

2.5.15 Test 16 - Error Flag and B-Code Verification Part III

Verifies that a write function to a nonexistant sector will produce an error and the correct B-code is produced. The deleted data bit will also be cleared.

NOTE

Test 16 must be run before test 17 as test 16 clears the deleted data bit and test 17 tests that it is cleared.

2.5.16 Test 17 - Illegal Track Error Verification

This test verifies that if a track address larger than 114(octal) is accessed, an error condition will occur, and the B-code will = 40. It also expects the deleted data bit to be cleared.

2.5.17 Test 20 - Seek Verification Via Read Function

This test does a read function on the selected tracks testing for seek errors on various sections of the diskette.

2.5.18 Test 21 - Write Test

The purpose of this test is to write all ones on sector 1, track 1, and to verify that the data in the sector buffer is not changed.

NOTE

This test must be run before tests 22 & 23 as they check for data written on track 1 sector 1.

2.5.19 Test 22 - Initialize Implied Read

After previously writing data on track 1 sector 1, this test changes the contents of the sector buffer and does a programmed initialize. At the end of an init. (recal.) the sector buffer must contain the data from track 1 sector 1.

NOTE

Unit 0 must be on-line for this test to work.

2.5.20 Test 23 - Read Test

This test verifies that a read function does infact load the sector buffer with data read from the selected address.

2.5.21 Test 24 - Data Transfer and Verification

The purpose of this test is to write then read and check data on all sectors of the selected tracks. The test alternates between drives, if both drives are selected, before changing tracks. The data pattern used is a floating 0 pattern.

2.5.22 Test 25 - Data Verification Via Deleted Data Mode.

This test is the same as Test 24 except it checks for deleted data indicators and uses a data pattern of floating 1.

2.5.23 Test 26 - Head "Home" Test

This test checks for the "home found before the desired track was reached" error code. The head is moved out 10 tracks then decremented back to track 0. It tests all selected drives, and uses a data pattern of random data.

3.0 ERRORS

Pretest and tests 1 - 17, and tests 21 - 23 handle errors as indicated in Section 3.1. For the most part these tests do not rely on an interrupt to indicate the function is completed. Whereas the other tests (tests 20, and 24 - 26) do read, write and read check functions over the selected track, sectors, and drives. These require the interrupt service and error detection that was used in the data reliability test. This is described in Section 3.3.

NOTE

If loop on error switch is up then the program will loop on the shortest set of instructions that will keep it in the failing loop. Otherwise after reporting the error the program will continue running through the remaining addresses and tests.

3.1 Error Heading For Tests 1 - 17, And 21 - 23 Plus Pretest.

The error heading is as follows:

ERADR FAST FAPT [BLANK] GOOD BAD

Under each column the error routine prints pertinent information.

ERADR = Error address
Address of the error trap instruction where the error was detected.

FAST = First address of selected test
Address of the test selected and running

FAPT = First address of present test
Address of the test or subtest presently running, or address of the scope loop.

[BLANK]
Additional general information supplied by some tests on an error.

GOOD = Expected results of the test
Test results of what should have happened if there was no error.

BAD = Actual test results
The data that was received from the RX01, that caused the error.

PASS = Number of passes made up to this error

3.2 Error Output Per Test

The following are the types of print outs under the columns [blank], good, and bad for the various tests, using this errror format.

TEST (SECTION)	[BLANK] (R2)	GOOD (R0)	BAD (R1)
PRETEST (1)	N/A	40	(RXCS)
PRETEST (2)	(RXCS) INCL.DD BIT	4 OR 204	(RXCS) NO DD BIT
TEST 1 (1)	N/A	40	(RXCS)
TEST 1 (2)	N/A	0	(RXCS)
TEST 1 (3)	(KRXVEC)	N/A	N/A
TEST 2 (1)	(KRXVEC)	N/A	N/A
TEST 2 (2)	(KRXVEC)	140	(RXCS)
TEST 2 (3)	(KRXVEC)	40	(RXCS)
TEST 2 (4)	(KRXVEC)	40	(RXCS)
TEST 2 (5)	(KRXVEC)	40	(RXCS)
TEST 3 (1)	(KRXVEC)	N/A	N/A
TEST 4 (1)	(KRXVEC)	N/A	N/A
TEST 5 (1)	N/A	40	(RXCS)
TEST 5 (2)	(RXDB) INCL.DD BIT	4 OR 204	(RXDB) NO DD BIT
TEST 5 (3)	N/A	0	(RXCS)
TEST 5 (4)	N/A	40	(RXCS)
TEST 5 (5)	(RXCS) INCL.DD BIT	200	(RXCS) NO DD BIT
TEST 6 (1)	NO. OF XFERS	N/A	N/A
TEST 7 (1)	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA

TEST 10 (1)	NO. OF XFERS	EXPEC. DATA	ACTUAL DATA
TEST 11&12 (1)	[USES TEST 6 & 7 TO FILL / EMPTY BUFFER]		
TEST 13 (1)	(RXDB)	200	(RXDB) NO DD BIT
TEST 13 (2)	(RXDB)	200	(RXDB) NO DD BIT
TEST 14 (1)	NO. OF TR'S	100040	(RXCS)
TEST 14 (2)	(RXDB)	0	(RXDB) NO DD BIT
TEST 14 (3)	(RXDB)	40	(RXCS)
TEST 14 (4)	N/A	70	(RXDB) ERROR CODE
TEST 15 (1)	NO. OF TR'S	100040	(RXCS)
TEST 15 (2)	N/A	100	(RXDB)
TEST 15 (3)	N/A	70	(RXDB) ERROR CODE
TEST 16 (1)	NO. OF TR'S	100040	(RXCS)
TEST 16 (2)	N/A	0	(RXDB)
TEST 16 (3)	N/A	70	(RXDB) ERROR CODE
TEST 17 (1A)	(RXDB)	0	(RXCS)
TEST 17 (1B)	N/A	100040	(RXCS)
TEST 17 (2)	N/A	0	(RXDB)
TEST 17 (3)	(RXDB)	40	(RXCS)
TEST 17 (4)	N/A	40	(RXDB) ERROR CODE
TEST 21 (1)	(RXES) STATUS A	NO. OF BYTE	(RXDB) STATUS B
TEST 21 (2)	[USES TEST 7 TO EMPTY BUFFER]		
TEST 22	[USES TEST 6 & 7 TO FILL AND EMPTY BUFFER]		
TEST 23	[USES TEST 6 & 21 TO FILL AND CHECK BUFFER]		

3.3 Error Heading For Tests 20, 24 - 26

As previously stated these tests access all the selected sectors, tracks, and drives, and rely on the interrupt service routine to indicate that a function is completed or an error occurred. All errors, with the exceptions where noted, will type as its first or second line of the message "error conditions test PC = XXXX PASS = x". The test PC number is the starting address of the test running, and the pass number is the number of passes made up to the error.

On most errors the program will type out the contents of "Status A" and "Status B".

Status A is the contents of the RXES (error and status register) at the time the error was detected. It shows the CRC, PAR, etc. errors.

Status B is the "definitive error codes" that the RX01 detected, that may have caused the error condition. These error codes are defined in Section 3.3.4

There are three categories of errors as listed and described below.

3.3.1 No Error Flag Errors

These are errors that can occur but the error flag in the RXCS will not be set.

1. Unexpected or missing deleted data bit

   This error results when the program expects and doesn't see the DD bit ("D D mark missing"), or doesn't expect and finds the deleted data bit set ("unexpected D D mark"). The program will type out at what diskette address this occurred then continue testing.

   NOTE

   See Section 3.3.3 for other causes of this error.

2. Data no status error

   This error occurs during a read check when the data read does not match the data in the memory data buffer, and there was no CRC error indicated. This means that the data was probably read off the diskette correctly but the transfer between the sector buffer and the RXDB in the RX11 produced bad data.

The error message will include the diskette address, "byte" number in the sector, the data read from the sector buffer "bad", and the expected data from the memory buffer "good".

Byte # Bad Good
(The data patterns are formatted as shown)

0 (Track address; bits 6 - 0)
1 (Unit number bit 7)
(Sector address bits 4 - 0)

Bytes 2 - 125 contain the selected data pattern.

126 (The sum of all bytes 0 - 125)
127 (The negative of 2 times byte 125)

The program detects a checksum errror by summing all the data read from the sector buffer and comparing that sum to 0.

At the end of the data error typeout the program prints if the checksum accumulated was "good" or "bad". If bytes 0 or 1 have data errors the operator must check the results of the checksum. If it is also bad, then there was a true data error. If the checksum was good, then it might be that the head is not over the track expected, and there is a positioning error.

If switch 9 is down then only 10 data errors will be printed, and at the end of the sector the "total read check errors =" will be typed. If switch 9 is up then all the data errors for that sector will be typed out.

3. Power failure

The program tests for two types of power failure, total system power loss, and RX11 power loss resulting in a recalibration of the drives.

The total system power failure is detected by "SYSMAC" subroutine ".$POWER". When the power is detected to be going down, the registers are saved. When the power comes back up the registers are restored and the message "power" is printed. The program then restarts.

Loss of power in the RX11 causes a recalibration of all drives. When this happens the "init done" bit is set in the RXES register along with the normal done flag At each interrupt the program tests for the init done bit. If it is found set, the function was not completed and a power loss must have been detected. When this happpens the program types out "RX11 Power" and restarts. The error heading is not typed on this error.

4. Unknown interrupt

   If an interrupt occurs through the RX11 interrupt vector address and none of the status bits are set (done, error, etc.) the program will type "unknown interrupt" and return back to the program to continue the function. The error heading is not printed.

5. No interrupt at done

   The program expects an interrupt at done on the functions of these tests. if an interrupt does not occur at done time then the program will type out "no interrupt at done error" then go into the interrupt service routine as if an interrupt did occur. At this point other errors may be printed if any are detected.

3.3.2 Error Flag Errors

These errors are detected as the results of the error bit being set in the RXCS at an interrupt.

1. Parity error

   A parity error results from an incorrect transfer of a command word from the RX11 interface to the RX01 micro-processor controller. The program will type out the contents of the command status register (RXCS) showing the function that failed, the address of the error, contents of status A (RXES) with the parity bit set, contents of status B (RXDB) with the definitive error code of 210 set. Then a "read, write, fill buffer or empty buffer parity error" will be printed. If a parity error occurs on a "read definitive error code" function, then the contents of the RXCS and "parity error" will be typed out.

2. CRC errors

   On all data transfers between the sector buffer and the diskette, a CRC word is generated and checked. If an error is detected by the micro-processor in this CRC word then a CRC error is generated. The program again types out the contents of the registers (RXCS contains function, status A with "CRC ERR" bit set, status B with an error code of 200). Then if it is a read only function, or a read check function and there were data errors it will type out "data CRC errors" then print the bad bytes if any. If it was a read check function and there were no data errors it will print "CRC error no data error".

3. Seek errors

Any error that produces a definitive error code but does not set an error bit in status A (RXDB at end of function) is labeled a seek error. See Section 3.3.4 for error codes and meanings. The same information is printed for these errors as in parity, or CRC errors, except it states that it is a "write or read seek error". If switch 8 is down then at each seek error found the program does an initialize of the RX01 so it will recalibrate to a known (home) position. The program then goes on to the next sector or track and continues testing, if the loop on error switch is off. (See Section 3.3.3. for errors caused by previous errors.) If the loop on error switch is up it will retry the function at the same address. If switch 8 is up then no "initialize" is done and the program looks at the other switches for operating conditions. Seek errors also print the track address that the head moved from at the time of the error.

4. Error flag error

If the error flag is not set in the RXCS and an error bit is set in status A or an error code is set in status B then there was an error but the error flag was not set. The message "error flag error" is printed then the program continues to type out the type of error.

3.3.3 Errors Resulting From Previous Errors

If there is a "write seek error" the program will go on to the next address without writing on the address where the error occurred. (Unless the loop on error switch 11 is up and the seek error is recovered.) If the write function is followed by a read check function and the read does not have a seek error at the same address, then there may be data errors, or unexpected or missing deleted data bit errors resulting from no data being written on that address by the previous write function.

3.3.4 Definitive Error Codes

The RX01 micro-processor has defined the error codes and meanings which are available to the program by issuing command #7 "read definitive error code"
The following are the codes and their meanings:

10 - Drive 0 failed to see home from initialize
20 - Drive 1 failed to see home from initialize
30 - Home found when stepping out 10 tracks for init.
40 - Tried to access a track greater then 76
50 - Home found before desired track was reached
60 - Self diagnostic error
70 - Desired sector not found after sampling 52 headers
100 - Write protect error
110 - More then 40 us and no sep clock detected
120 - A preamble could not be found
130 - Preamble found but no ID mark found in time
140 - CRC error on a header, no error flag
150 - Good header (no CRC error) but track compare error
160 - ID address mark not found in time
170 - Data mark not found in time
200 - Data CRC error
210 - Parity errors

3.4 Program Hung

If there is no response from the RX11 while waiting for the transfer request (TR) flag or the done flag, the program will type "device test hung @ PC" (only if SW13 is off) and then go on to the next test, or the beginning of the present test.

4.0 HALTS

The only halts in the program are the selectable halts (EOP, EOT, at error), the illegal vector halts, and the illegal test selection halt.

NOTE

One additional 'halt' exists in the program. It occurs when the user has loaded his program via the 'RXDP' monitor (on unit 0) and also requires testing of unit 0. A prompt message is typed reminding the user to replace his load medium with a scratch diskette before going on. The program will wait for the 'continue' switch to be depressed.

SEQ 0028

5.0 FLOW CHARTS

```
                                        .NLIST CND,MD,MC
                                        .LIST ME
                                        .ENABL ABS,AMA


                                .TITLE  CZRXBF0 RX11 INTERFACE TEST
                                ;*COPYRIGHT (C) MAY 8,1979
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY D. ADAMS/B. BURGESS/MIKE PAGE
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
000001                          $TN=1
160000                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT


                                        ;COPYRIGHT (C) 1975,1976
                                        ;THIS SOFTWARE IS FURNISHED UNDER LICENCE FOR USE ONLY
                                        ;ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
                                        ;THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS
                                        ;SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
                                        ;OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                                        ;EXCEPT FOR USE ON SUCH SYSTEM, AND TO ONE WHO AGREES TO
                                        ;THESE LICENCE TERMS. TITLE TO OWNERSHIP OF THE
                                        ;SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

                                        ;THE INFOMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
                                        ;WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
                                        ;BY DIGITAL EQUIPMENT CORPORATION.

                                        ;DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
                                        ;OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
```

```
                        ;MODIFIED TO REV. D BY B. BURGESS NOV. 10, 1975 AS FOLLOWS:
                        ;
                        ;A)     ADDED CAPABILITY OF VARIABLE DEVICE 'BR' LEVEL.  ALL RELEVANT TESTS
                        ;       CALCULATE 'CPU' LEVEL BASED ON CURRENT CONTENTS OF LOCATION 'BRLEV:'.
                        ;       DEFAULT 'BR' LEVEL, FOR THE DEVICE, SET BY THE PROGRAM IS 5.  ANY OTHER
                        ;       'BR' LEVEL ( E.G. 6 ) WOULD HAVE TO BE PATCHED INTO LOCATION 'BRLEV:'
                        ;       BEFORE RUNNING THE PROGRAM.
                        ;
                        ;B)     ADDED TWO (2) ROUTINES TO HANDLE 'UNEXPECTED' BUS TIMEOUT AND
                        ;       RESERVED INSTRUCTION TRAPS ( TRAPS TO VECTORS 4 & 10, RESPECTIVELY).
                        ;       BOTH ROUTINES WILL INDICATE WHICH TRAP OCCURRED, THE 'PC' LOCATION
                        ;       OF WHERE THE TRAP OCCURRED, AND ATTEMPT TO RESTART THE PROGRAM.
                        ;
                        ;C)     ADDED CODE TO FAILSAFE UNIT 0 UNDERGOING TESTING IF PROGRAM WAS
                        ;       LOADED VIA UNIT 0 USING 'RXDP' MONITOR AND USER STARTED RUNNING
                        ;       THE PROGRAM WITHOUT HAVING REPLACED HIS LOAD MEDIUM WITH A 'SCRATCH'
                        ;       DISKETTE.
                        ;
                        ;D)     ADDED MESSAGES TO INDICATE TO USER WHEN HE HAS SELECTED TRACK AND/OR
                        ;       SECTOR LIMITS 'OUT OF RANGE' AND CORRESPONDING DEFAULT LIMITS WHEN
                        ;       THIS CONDITION ARISES
                        ;
                        ;E)     MODIFIED TESTS 1 THRU 4 TO CORRECTLY PRINT OUT THE CONTENTS OF
                        ;       'KRXVEC' ( LOCATION HOLDING THE DEVICE VECTOR) AS 264 INSTEAD OF 270.
                        ;
                        ;F)     MODIFIED TEST 2 TO HANDLE A 'LOCKED IN INTERRUPT STATE' CONDITION
                        ;       ARISING WHEN 'INTERRUPT ENABLE' AND 'DONE' ARE BOTH QUALIFIED AND
                        ;       THE 'REQUEST INTERRUPT' FLOP NEVER GETS CLEARED.
                        ;
                        ;G)     ADDED EXTENSIVE MAINTENANCE INFORMATION BASED ON FAULT INSERTION
                        ;       RESULTS.  INFORMATION IS KEYED TO THE 'ERROR' REPORT WITHIN A
                        ;       TEST.  INFORMATION PROVIDED SHOULD BE SELF-EXPLANATORY BUT SHOULD
                        ;       NOT BE MISCONSTRUED AS BEING ALL ENCOMPASSING DUE TO HUMAN ERRORS
                        ;       IN STATISTICS GATHERING, INABILITY TO FAULT INSERT SOME CHIPS, AS
                        ;       WELL AS ONLY TWO (2) MODULES ABLE TO BE FAULT INSERTED I.E. -
                        ;       M7846 (UNIBUS INTERFACE) AND M7727 (READ/WRITE CONTROL).
                        ;
                        ;H)     ADDED FLOW CHARTS
                        ;
```

```
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
001200                  STACK=  1200
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
```

```
                        SW02=   4
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
```

```
                                        ;SPECIAL EQUATES


            000017                      RDER    =17             ; READ B CODE
            000040                      DONEBIT =40
            000101                      FBIE    =101            ; IE+FILL BUFFER
            000103                      EBIE    =103            ; IE+EMPTY BUFFER
            000105                      WRTIE   =105            ; IE+WRITE SECTOR
            000107                      RDIE    =107            ; IE+READ SECTOR
            000115                      WTDDIE  =115            ; IE+WRITE DD SECTOR
            040001                      RECAL   =40001
            000000                      OPEN    =0

            000000                      .=0
000000      000000  000000                      .WORD 0,0

            000004                      .=4
000004      006156                      .WORD   BUSERR          ;UNEXPECTED TIMEOUT TRAP PC
000006      000340                      .WORD   PR7             ;UNEXPECTED TIMEOUT TRAP PS
000010      006202                      .WORD   RESERR          ;UNEXPECTED RESERVED INSTRUCTION TRAP PC
000012      000340                      .WORD   PR7             ;UNEXPECTED RESERVED INSTRUCTION TRAP PS

            000020                      .=20
000020      006524                      XSCOPE
000022      000340                      PR7
000024      015160                      $PWRDN
000026      000340                      PR7
000030      006260                      XERROR
000032      000340                      PR7
000034      015074                      $TRAP                   ;ADDRESS OF TRAP SERVICE
000036      000340                      PR7


            000042                      .=42
000042      000000                      .WORD   0

            000046                      .=46
000046      002470                      LOGICAL                 ;ACT 11 EOP HOOKS


            000052                      .=52
000052      000000                      .WORD 0

            000174                      .=174
000174      000000                      DISPREG:        0
000176      000000                      SWREG:          0


            000200                      .=200
000200      000401                              BR 1$
000202      000402                              BR 2$
000204      000137  001232          1$:         JMP SA200           ;OPERATOR SELECTED CONDITIONS
000210      000137  001222          2$:         JMP SA202           ;RESTART PROGRAM WITH PREVIOUS PARAMETERS
```

```
248
249
```

```
                        .SBTTL TEST SELECTION VIA SWITCH REGISTER

                ;;**************************************************************
                ;;**************************************************************
                ;;**************************************************************

                        ;SET TEST AND DRIVE SELECTION IN " DTESTP " LOCATION 1212
                        ;
                        ; BIT 15 = 1 - UNIT 1 SELECTED
                        ; BIT 14 = 1 - UNIT 0 SELECTED
                        ; BIT 15 & BIT 14 = 0 - BOTH DRIVES MUST BE READY
                        ;
                        ; BIT 4 - BIT 0 = OCTAL NUMBER OF DESIRED STARTING TEST
                        ; BIT 4 - BIT 0 = 0 -ALL TESTS WILL BE SEQUENCED THROUGH

                ;;**************************************************************
                ;;**************************************************************
                ;;**************************************************************


                        .SBTTL OPERATIONAL SWITCH REGISTER POSITIONS

                ;;**************************************************************
                ;;**************************************************************
                ;;**************************************************************

                ; SET OPERATING CONDITIONS IN THE SWITCH REGISTER (HARDWARE)
                ; OR SOFTWARE SWITCH REGISTER LOCATION 176

                ; 15 = 1 - HALT ON ERROR
                ; 14 = 1 - HALT AT END OF PASS
                ; 13 = 1 - INHIBIT ERROR TYPEOUT
                ; 12 = 1 - LOOP ON TEST
                ; 11 = 1 - LOCK ON ERROR
                ; 10 = 1 - HALT AT END OF TEST
                ;  9 = 1 - PRINT ALL DATA ERRORS
                ;  9 = 0 - PRINT ONLY FIRST 10 DATA ERRORS PER SECTOR
                ;  8 = 1 - INHIBIT RECALIBRATION ON SEEK ERRORS.
                ;
                ;  0 = 1 - INHIBIT <BELL> AT ERROR

                ; 15-0 = 1 - SELECT SOFTWARE SWITCH REGISTER

                ;;**************************************************************
                ;;**************************************************************
                ;;**************************************************************
```

```
                                .SBTTL RXCS (RX COMMAND STATUS REGISTER)

001200                          .=STACK

001200  000000          OD:     0                       ;OD/ID = 0 UNLESS SPECIFIC TRACKS SELECTED.
001201                  ID=OD+1
001202  015001          FIRST:  015001                  ; FIRST = 1, LAST = 32
001203                  LAST=FIRST+1

001204  000264          KRXVEC: 264

001206  177170          RXCS:   177170

                        ; RXCS: STANDARD DEVICE ADDRESS = 177170

                        ; TOGGLE INTO PROGRAM LOCATION " RXCS " THE RX11 DEVICE ADDRESS IF NOT = 177170

                        ;KEY: R - READ ONLY BIT
                        ;     W - WRITE ONLY BIT

                        ;               15 - R  - ERROR
                        ;               14 - W  - INITIALIZE
                        ;               13 -
                        ;               12 -
                        ;               11 -    (BITS 13-8)
                        ;               10 -    (NOT USED)
                        ;                9 -
                        ;                8 -
                        ;                7 - R  - TRANSFER REQUEST
                        ;                6 - R/W- INTERRUPT ENABLE
                        ;                5 - R  - DONE
                        ;                4 - W  - UNIT SELECT
                        ;                3 - W  - FUNCTION
                        ;                2 - W  - FUNCTION
                        ;                1 - W  - FUNCTION
                        ;                0 - W  - GO !

                        ; FUNCTION

                        ; 3 2 1 0

                        ; - - - GO

                        ;   0 + GO -    FILL BUFFER
                        ;   2 + GO -    EMPTY BUFFER
                        ;   4 + GO -    WRITE SECTOR
                        ;   6 + GO -    READ SECTOR
                        ;   -
                        ;  12 + GO -    READ STATUS " A "
                        ;  14 + GO -    WRITE DELETED DATA
                        ;  16 + GO -    READ STATUS " B " (CODES)
```

```
                                        .SBTTL RXDB (RX DATA BUFFER REGISTER)

001210  177172                  RXDB:   177172

                                ; RXDB: STANDARD DEVICE ADRESS = 177172

                                ; THE FOLLOWING BIT IDENTIFICATION REPRESENTS THE STATUS AT THE END OF A FUNCTION
                                ; (BUT NOT FUNCTION # 16 TO READ STATUS " B ") DISPLAYED WITHIN THE RX-DATA BUFFER.

                                ;       (A)     7 - SELECTED DRIVE READY
                                ;               6 - DELETED DATA
                                ;               5 -
                                ;               4 -
                                ;               3 - WRITE PROTECT ERROR
                                ;       (B)     2 - INITIALIZE DONE
                                ;               1 - PARITY ERROR
                                ;               0 - CRC ERROR

                                ; (A) - VISIBLE ONLY IF THE FUNCTION WAS # 12 READ STATUS " A "
                                ;
                                ; (B) - INIT DONE VISIBLE IF AN INITIZLIAE [KEY] OR [PROGRAMMED] WAS ISSUED

001212  000000                  DTESTP: 0

001214  000005                  BRLEV:  5

                                ;BRLEV: STANDARD PRIORITY INTERRUPT LEVEL = 5

                                ;TOGGLE INTO PROGRAM LOCATION "BRLEV" THE RX11 INTERRUPT PRIORITY
                                ;LEVEL IF NOT = 5

001216  177570                  SWR:            .WORD DSWR
001220  177570                  DISPLAY:        .WORD DDISP

                                ; R0 - GOOD     /EXPECTED RESULT OF TEST
                                ; R1 - EAC      /ACTUAL RESULT OF TEST
                                ; R2 - BLANK    /
                                ; R3 - TEST Q

                                ;;*****************************************************************

                                ;WORD "UNITSEL" HAS THE FOLLOWING BIT DEFFINITIONS
                                ;
                                ;BIT15 = 1 - UNIT 1 SELECTED FOR USE
                                ;BIT14 = 1 - UNIT 1 IN USE
                                ;BIT8  = 1 - THIS PASS HAD AN ERROR
                                ;BIT7  = 1 - UNIT 0 SELECTED FOR USE
                                ;BIT6  = 1 - UNIT 0 IN USE
                                ;BIT4  =     UNIT SELECTION BIT

                                ;;*****************************************************************
```

```
                                        .SBTTL START AND RESTART ADDRESSES

                                ; THE STARTING ADDRESS WAS 202

001222  005200                  SA202:  INC R0
001224  012706  001200                  MOV #STACK,SP
001230  000447                          BR RESTART

                                ; THE STARTING ADDRESS WAS 200

001232  005000                  SA200:  CLR R0
001234  012737  177570  001216          MOV #177570,SWR          ;RESET TO HARDWARE SWR.
001242  012706  001200                  MOV #STACK,SP
001246  104401  016746                  TYPE ,MREV               ;PRINT THE NAME AND REVISION
001252  013746  000004                  MOV 4,-(SP)              ;SAVE 'BUSERR' TIMEOUT 'PC'
001256  012737  001276  000004          MOV #1$,4                ;SET UP TIMEOUT VECTOR
001264  022777  177777  177724          CMP #177777,@SWR         ;IS SOFTWARE SWR SELECTED
001272  001402                          BEQ 2$                   ;YES, INSERT IT'S ADDRESS
001274  000423                          BR 3$                    ;BR IF NO TIMEOUT TRAP OCCURS
001276  022626                  1$:     CMP (SP)+,(SP)+          ;RESTORE THE STACK
001300  012737  000176  001216  2$:     MOV #SWREG,SWR           ;POINT TO SOFTWARE SWITCH REGISTER
001306  012737  000174  001220          MOV #DISPREG,DISPLAY     ;POINT TO SOFTWARE DISPLAY REG.

                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001314  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001320  001006                          BNE     64$             ;;BRANCH IF YES
001322  023727  001216  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001330  001005                          BNE     65$             ;;BRANCH IF NO
001332  104405                          GTSWR                   ;;GET SOFT-SWR SETTINGS
001334  000403                          BR      65$
001336  112737  000001  015072  64$:     MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
001344                          65$:
001344  012637  000004          3$:     MOV (SP)+,4                     ;RESET TIMEOUT VECTOR TO 'BUSERR'
001350  000005                  RESTART:RESET                           ;INITIALIZE THE RX11 SYSTEM
001352  012746  000340                  MOV #PR7,-(SP)
001356  012746  001364                  MOV #4$,-(SP)
001362  000002                          RTI                             ;LOAD THE PSW
001364  013737  001206  001210  4$:     MOV RXCS, RXDB                  ;GET ADDRESS OF RXCS
001372  062737  000002  001210          ADD #2, RXDB                    ;SET UP ADDRESS OF RXDB
001400  012737  001234  012654          MOV #001234,RAN1                ; INITIALIZE CONSTANTS OF
001406  012737  000765  012656          MOV #000765,RAN2                ; RANDOM NUMBER GENERATOR
001414  005037  002554                  CLR CCOUNT
001420  005037  002556                  CLR PASS
001424  005037  007004                  CLR HANGER
001430  012737  177740  007006          MOV #177740,HANGPL
001436  005700                          TST R0
001440  001064                          BNE XSA202                      ; STARTING ADDRESS WAS 202
001442  005037  012752                  CLR UNITSEL
001446  032737  140000  001212          BIT #140000,DTESTP              ;WERE ANY DRIVES SELECTED
001454  001004                          BNE 1$                          ;YES GO SET THEIR BITS
001456  052737  100200  012752          BIS #100200,UNITSEL             ;NO, BOTH UNITS MUST BE READY
001464  000415                          BR 2$
001466  032737  040000  001212  1$:     BIT #BIT14,DTESTP               ;WAS UNIT 0 SELECTED
001474  001443                          BEQ 3$                          ;NO, MUST BE UNIT 1
001476  052737  000200  012752          BIS #200,UNITSEL                ;YES,SET SELECTED BIT
001504  005737  001212                  TST DTESTP                      ;WAS UNIT 1 SELECTED
001510  100003                          BPL 2$                          ;NO
```

```
                                        BIS #BIT15,UNITSEL              ;YES,SET THE SELECTED BIT
                                2$:     TST     @#42                    ;AUTO MODE?
                                        BEQ     XSA202                  ;BRANCH IF NOT
                                        CMP     @#42,@#46               ;ACT MODE ?
                                        BNE     6$                      ;BRANCH IF NOT
                                        MOV     #SWREG,@SWR             ;GET SOFT SWITCH REG FOR AUTO MODE
                                        BIS     #BIT15,@SWR             ;SET FOR HALT ON ERROR
                                        BR      XSA202
                                6$:     BITB    #10,@#41                ;WAS PROG. LOADED FROM RX01 ?
                                        BEQ     XSA202                  ;BRANCH IF NOT
                                        BIC     #200,UNITSEL
                                        TYPE    ,MUNIT1
                                        TYPE    ,MONLY
                                        BR      XSA202
                                3$:     BIS     #BIT15,UNITSEL
                                XSA202: BIC     #100200,OD              ;CLEAR 1ST TIME BITS FOR BOTH DRIVES
                                        TYPE, MDTESTP
                                        MOV     DTESTP,-(SP)    ;;SAVE DTESTP FOR TYPEOUT
                                        TYPOS                   ;;GO TYPE--OCTAL ASCII
                                        .BYTE   6               ;;TYPE 6 DIGIT(S)
                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                        TYPE ,MCRLF
                                LIMITS: TST OD
                                        BNE TRKLMT
                                        CLR SEQUEN
                                        TYPE ,MLIMTRK
                                        BR SECLMT

                                ; 0 <= OD <= ID <= 114

                                TRKLMT: CMPB ID,#114
                                        BHI 1$
                                        CMPB OD,ID
                                        BHI 1$
                                        TYPE ,MOD
                                        MOVB OD,-(SP)
                                        TYPOS
                                        .BYTE  3
                                        .BYTE  0
                                        TYPE ,MID
                                        MOVB ID,-(SP)
                                        TYPOS
                                        .BYTE  3
                                        .BYTE  0
                                        BR SECLMT
                                1$:     TYPE,   OD2BIG                  ;TYPE MSG. INDICATING ID OR OD
                                                                        ;TOO BIG & DEFAULTING TO TRACKS
                                                                        ;0, 52, 53, 114
                                        CLR OD
                                        BR LIMITS

                                ; 1 <= FIRST <= LAST <= 32

                                SECLMT: TSTB FIRST
                                        BNE 1$
                                        MOVB #1,FIRST
```

```
 511  001760  123727  001203  000032  1$:     CMPB LAST,#32
 512  001766  101023                          BHI 2$
 513  001770  123737  001202  001203          CMPB FIRST,LAST
 514  001776  101017                          BHI 2$
 515  002000  104401  016521          3$:     TYPE ,MFIRST
 516  002004  113746  001202                  MOVB FIRST,-(SP)
 517  002010  104403                          TYPOS
 518  002012     003                          .BYTE  3
 519  002013     000                          .BYTE  0
 520  002014  104401  016532                  TYPE ,MLAST
 521  002020  113746  001203                  MOVB LAST,-(SP)
 522  002024  104403                          TYPOS
 523  002026     003                          .BYTE  3
 524  002027     000                          .BYTE  0
 525  002030  104401  016162                  TYPE ,MCRLF
 526  002034  000406                          BR PRETEST
 527  002036  104401  017213          2$:     TYPE,   S2BIG                   ;TYPE MSG. INDICATING THAT
 528                                                                          ;SECTOR RANGE SELECTED WAS
 529                                                                          ;INVALID AND DEFAULTING TO A
 530  002042  012737  015001  001202          MOV #15001,FIRST                ; 1ST SECTOR VALUE OF 1 AND
 531  002050  000753                          BR 3$                           ;A LAST SECTOR VALUE OF 32
```

```
                                                .SBTTL PRETEST - INITIALIZE [KEY] PART I
                                ;
                                ;       "KEY" INITIALIZE SHOULD HAVE [SET] THE DONE FLAG BECAUSE
                                ;       ANY [INIT] OF THE RX01 MICROCONTROLLER IS AN IMPLIED [READ SECTOR]
                                ;       OF TRACK 1 SECTOR 1 (FOR SYSTEMS PROGRAMMING BOOTSTRAP APPLICATIONS).
                                ;
                                ;       THEREFORE, ANY ERROR (EXCEPT PARITY) THAT MAY OCCUR FROM A NORMAL
                                ;       "READ SECTOR" COMMAND MAY OCCUR HERE CAUSING THE ERROR FLAG TO SET, AND
                                ;       DISPLAYING THE ERROR STATUS WITHIN THE TRANSFER REGISTER AT "DONE".
                                ;
                                ;       THE TRANSFER REQUEST FLAG SHOULD BE CLEARED.
                                ;
                                ;NOTE:  SCOPE LOOPING IS NOT OFFERED BECAUSE THE "INIT" FUNCTION
                                ;       WHICH PRODUCED THE ERROR HAS NOT YET BEEN VERIFIED.

 002052  042737  000400  012752  PRETEST: BIC #BIT8,UNITSEL
 002060  005037  006520                   CLR ERRORS
 002064  000004                           SCOPE                       ; REFRESHES FAST FOR ERROR TYPEOUT
 002066  013737  006604  002560           MOV PCSCOPE,FAST            ; FOR ERROR TYPEOUT INFORMATION
 002074  012700  000040                   MOV #40, R0                 ; PROGRAM EXPECTS DONE FLAG
 002100  017701  177102          3$:      MOV @ RXCS, R1              ; ACTUAL CONTENTS OF RXCS
 002104  020100                           CMP R1, R0
 002106  001407                           BEQ 1$                      ; OK
 002110  062737  000001  007004           ADD #1,HANGER               ;WAIT FOR THE INIT. FUNCTION TO
 002116  005537  007006                   ADC HANGPL                  ;FINISH BEFORE CALLING IT AN ERROR
 002122  001366                           BNE 3$

                                 ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A

 002124  104000                           ERROR                       ; RXCS NOT = 40 (DONE)

      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

              ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
              ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
              ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
              ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
              ;
              ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
              ;ANALYZE THE FOLLOWING AREA/S:
      ;
      ;M7846 (UNIBUS INTERFACE)
      ;
      ;       SIGNAL NAME               REASON                   POSSIBLE CHIPS
      ;
      ;


      ;       NOTE:  MAKE SURE THE DRIVES ARE CONNECTED CORECTLY,THE
      ;                DISKETTES INSERTED,AND THE DOORS OF THE SELECTED DRIVES
      ;                ARE CLOSED. IF THE THESE CONDITIONS ARE NOT SET THERE
      ;                WILL BE AN ERROR AT THIS POINT.

      ;B TRANSFER REQUEST     STUCK HIGH                       E22
```

```
        ;DONE                   STUCK LOW                       E22,E36,E1
        ;RUN(0) H               STUCK LOW                       E22
        ;SELECT 00              STUCK LOW                       E18,E34,E17,E40,E11
        ;B DONE                 STUCK LOW                       E18,E15,E19,E41
        ;RUN(1) H               STUCK HIGH                      E18
        ;RX INIT                STUCK HIGH                      E32
        ;OUT                    STUCK HIGH                      E23,E4
        ;B INIT                 STUCK HIGH                      E36,E8
        ;BUS -> RXCS            STUCK HIGH/LOW                  E36,E40
        ;BUS INTR               STUCK HIGH                      E38
        ;BUS D02/D04            STUCK LOW                       E38
        ;RUN(1) H               STUCK LOW                       E37
        ;INT ENB(1) H           STUCK LOW                       E37
        ;TRANSFER REQUEST       STUCK HIGH                      E1
        ;INT ENB(1) H           STUCK HIGH                      E1
        ; ----------            STROBE DISABLED                 E1
        ; ----------            'A' INPUTS NOT SELECTED         E1
        ;CMD                    STUCK LOW                       E17,E21
        ;B DONE                 STUCK HIGH                      E15
        ;RUN(0) H               STUCK HIGH                      E15
        ;OUT                    STUCK LOW                       E24
        ;RX INIT                STUCK LOW                       E21,E11
        ;IN                     STUCK HIGH                      E21
        ;RX RUN                 STUCK LOW                       E14
        ; ----------            LOCKED IN 'PRESET' STATE        E2
        ; ----------            NO STROBE SIGNAL                E3
        ;BUS D15                STUCK LOW                       E9
        ;DATA                   STUCK HIGH                      E11
        ;

        ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
        ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
        ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
        ;MODULE M7727.
;
;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
;
;M7727 (READ/WRITE CONTROL)
;
;       SIGNAL NAME             REASON                          POSSIBLE CHIPS
;
; ----------                    PIN 15 NOT AT GROUND            E15
;SEL TRK 0                      STUCK LOW                       E15
;DK0 TRK 0                      STUCK LOW                       E15
;SEL DK0                        STUCK LOW                       E13
;DC LO                          STUCK LOW                       E13
;OUT                            STUCK HIGH                      E13
; ----------                    E18 CLOCK LOCKED HIGH/LOW       E14,E18
; ----------                    REPLACE E18                      ----------
;INIT                           STUCK LOW                       E18,E16
; ----------                    PIN 14 NOT +5V THRU 1K          E16
```

```
      ; ----------              'J,K' INPUTS TO E18 LOCKED       E17
      ;                            HIGH/LOW
      ; ----------              'J' INPUT TO E16 LOCKED HIGH     E17
      ;
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*



                                ;//////////////////////////////////////////////////////////////////////
                                ;  STATUS A [RXDB] AT "DONE"
                                ;          7       6       -       -       3       2       1       0          /
                                ;
                                ;          SEL                             WRITE   INIT    PAR     CRC        /
                                ;          DRIVE   DD                      PROTECT [DONE]                     /
                                ;          RDY                                             (N/A)              /
                                ;
                                ;//////////////////////////////////////////////////////////////////////

002126  012737  177740  007006  1$:     MOV #177740,HANGPL                      ;RESET HANG COUNTER
002134  012700  000204                  MOV #204,R0                             ;EXPECT INIT DONE AND UNIT 0 RDY
002140  017702  177044                  MOV @RXDB,R2
002144  010201                          MOV R2,R1
002146  042701  000100                  BIC #BIT6,R1                            ;CLEAR DELETED DATA BIT
002152  105737  012752                  TSTB UNITSEL                            ;WAS UNIT 0 SELECTED
002156  100404                          BMI 2$
002160  042701  000200                  BIC #BIT7,R1                            ;UNIT 0 WAS NOT SELECTED
002164  042700  000200                  BIC #BIT7,R0                            ;CLEAR THE DRIVE 0 RDY BITS
002170  020100                  2$:     CMP R1, R0
002172  001544                          BEQ REBEGIN

                                ; (R0) = 4 IF UNIT 0 IS NOT SELECTED, OR 204 IF UNIT 0 IS SELECTED
                                ; (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
                                ; (R2) = ACTUAL RXDB

002174  104000                          ERROR                           ; RXDB NOT = 4, OR 204



      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

              ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
              ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
              ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
              ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
              ;
              ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
              ;ANALYZE THE FOLLOWING AREA/S:
      ;
      ;M7846 (UNIBUS INTERFACE)
      ;
      ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
```

```
        ;

        ;        NOTE:  MAKE SURE THE DRIVES ARE CONNECTED CORECTLY,THE
        ;                 DISKETTES INSERTED,AND THE DOORS OF THE SELECTED DRIVES
        ;                 ARE CLOSED. IF THE THESE CONDITIONS ARE NOT SET THERE
        ;                 WILL BE AN ERROR AT THIS POINT.

        ;B SHIFT                  STUCK LOW/HIGH              E15,E19,E2
        ;B DONE                   STUCK HIGH                  E15
        ;TO RX01                  STUCK LOW/HIGH              E15
        ;LOAD                     STUCK HIGH                  E17,E18
        ;SELECT 00                STUCK HIGH                  E17,E18
        ;SELECT 02                STUCK LOW                   E17,E18,E34
        ;IN                       STUCK LOW                   E17,E21
        ;CMD                      STUCK LOW/HIGH              E21
        ;BUS D15                  STUCK HIGH                  E40
        ;RX INIT                  STUCK HIGH                  E32
        ;BUS D00 -> D03           STUCK HIGH                  E41,E7
        ;DATA                     STUCK LOW                   E18,E11
        ; ----------              INCORRECT SHIFT OUTPUTS     E5
        ; ----------              CAN'T SELECT 'B' INPUTS     E1
        ; ----------              CAN'T RESET                 E3
        ;B INIT                   STUCK LOW                   E34,E8
        ;RX BUSY                  STUCK LOW                   E34,E22,E19
        ;B DONE                   STUCK LOW                   E19
        ;BUS D05                  STUCK LOW                   E4
        ;B SER DATA               STUCK LOW/HIGH              E9
        ;INT ENB(1) H             STUCK HIGH                  E37
        ;

                ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
                ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
                ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
                ;MODULE M7727.
        ;
        ;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
        ;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
        ;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
        ;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
        ;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
        ;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
        ;
        ;M7727 (READ/WRITE CONTROL)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ;SEL WT PROT              STUCK LOW                   E4,E6,E15
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 753  002176  000542                              BR      REBEGIN
 754  002200  004737  002604          MORETESTS: JSR PC, LOCKUP
 755  002204  005723                  TONOTHERE: TST (R3)+               ;ADJUST R3 FOR NEXT TEST ADDRESS
```

```
002206  016337  002232  006604  FIRSTTEST: MOV TESTS(R3), PCSCOPE       ; EQUIVALENT TO " SCOPE "
002214  013737  006604  002560             MOV PCSCOPE,FAST             ;SAVE THE FIRST ADDRESS OF THE TEST
002222  012706  001200                     MOV #STACK,SP
002226  000173  002232                     JMP @TESTS(R3)
002232  002204  002646  002766  TESTS:     TONOTHERE, T1, T2, T3, T4, T5, T6, T7
002240  003310  003446  003612
002246  004036  004170
002252  004166  004254  004244             T10, T11, T12, T13, T14, T15, T16, T17
002260  004320  004444  004636
002266  004776  005244
002272  005510  005534  005720             T20, T21, T22, T23, T24, T25, T26, NOMORETESTS
002300  005776  006044  006076
002306  006110  002312

                                ; TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I

                                ; TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION

                                ; * TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I

                                ; * TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II

                                ; TEST 5 - INIT [PROGRAMMED] / RST

                                ; TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION

                                ; TEST 7 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART I

                                ; TEST 10 - EMPTY BUFFER TRANSFER LENGTH AND CONTENT VERIFICATION PART II

                                ; TEST 11 - FILL/EMPTY BUFFER ALL 0'S

                                ; TEST 12 - FILL/EMPTY BUFFER ALL 1'S

                                ; TEST 13 - DRIVE READY VERIFICATION FOR SELECTED DRIVES

                                ; TEST 14 - ERROR FLAG AND B - CODE VERIFICATION PART I

                                ; TEST 15 - ERROR FLAG AND B - CODE VERIFICATION PART II
                                ;         /DELETED DATA BIT SETS

                                ;TEST 16 - ERROR FLAG AND B - CODE VERIFICATION PART III
                                ;         /DELETED DATA BIT CLEARS

                                ;TEST 17 - ILLEGAL TRACK ERROR AND B - CODE VERIFICATION

                                ;TEST 20 - SEEK VERIFICATION VIA READ FUNCTION

                                ;TEST 21 - WRITE TEST

                                ;TEST 22 - INITIALIZE IMPLIED READ

                                ;TEST 23 - READ TEST

                                ;TEST 24 - DATA TRANSFER & VERIFICATION
```

```
                        ;TEST 25 - DATA TRANSFER & VERIFICATION VIA DELETED DATA MODE
                        ;TEST 26 - HEAD "HOME" TEST
                        ;THERE ARE NO MORE TESTS
                        ; * NOTE: ON PROCESSORS WITHOUT HARDWARE PROCESSOR STATUS WORDS (PSW)
                        ;       THESE TEST WILL NOT BE RUN.
```

```
                                        ;PRINT AN END OF PASS INDICATOR
                                        ;
                                        ; C - RX11/RX01 TEST PASS OK
                                        ; D - RX11/RX01 AND DRIVE TESTING OK
                                        ; - - AN ERROR OCCURRED (DURING C OR D)
                                        ;
                                        ; NOTE: IF BIT 8 OF UNITSEL IS A 1
                                        ; THEN AN ERROR HAS OCCURRED FOR THIS PASS
                                        ;
002312  042777  000100  176666  NOMORETESTS:    BIC     #BIT6,@RXCS     ;CLEAR 'IE' BIT BEFORE NEXT PASS
002320  005037  007004                          CLR HANGER
002324  032737  000400  012752                  BIT #BIT8,UNITSEL               ;"C" OR "D" MEANS ERRORLESS PASS.
002332  001403                                  BEQ 1$
002334  012737  000055  002562                  MOV #'-, MX                     ; " - " MEANS UN-ERRORLESS PASS
002342  005737  002554          1$:             TST CCOUNT
002346  001002                                  BNE 3$
002350  104401  016162                          TYPE, MCRLF
002354  005237  002554          3$:             INC CCOUNT
002360  022737  000110  002554                  CMP #72., CCOUNT
002366  001002                                  BNE 4$
002370  005037  002554                          CLR CCOUNT
002374  104401  002562          4$:             TYPE, MX
002400  104401  006516                          TYPE ,MABELL
002404  005237  002556          2$:             INC PASS
002410  102775                                  BVS 2$
002412  104406                                  CKSWR
002414  032777  040000  176574                  BIT #SW14,@SWR                  ; AC SW 14 = 1 TO HALT AT END OF PASS
002422  001413                                  BEQ 6$
002424  104401  016162                          TYPE ,MCRLF
002430  104401  006753                          TYPE ,MPASS
002434  013737  002556  002446                  MOV PASS,5$
002442  004537  015642                          JSR R5,SGLDEC
002446  000000          5$:             OPEN
002450  000000                          HALT
002452  005237  007004          6$:             INC HANGER                      ;WAIT FOR EOP INDICATOR TO BE PRINTED
002456  001375                                  BNE 6$
002460  013705  000042                          MOV @#42,R5                     ;ACT 11 END OF PASS HOOKS
002464  001405                                  BEQ HERE
002466  000005                                  RESET
002470  004715          LOGICAL: JSR PC,(R5)
002472  000240                          NOP
002474  000240                          NOP
002476  000240                          NOP
002500  000137  002504          HERE:   JMP REBEGIN
                                        ;
002504  042737  000400  012752  REBEGIN: BIC #BIT8,UNITSEL                     ;CLEAR HARD ERROR INDICATOR
002512  013703  001212                          MOV DTESTP, R3
002516  042703  177740                          BIC #177740, R3                 ; R3 CONTAINS TEST # 0 TO 26
002522  020327  000027                          CMP R3,#27
002526  103006                                  BHIS 1$
002530  006303                                  ASL R3
002532  032777  000040  176446  2$:             BIT     #40,@RXCS
002540  001774                                  BEQ     2$
002542  000621                                  BR FIRSTTEST
                                        ;
002544  104401  002564          1$:             TYPE ,MILTST
```

```
   876  002550  000137  001232                    JMP SA200
   877
   878  002554  000000                    CCOUNT: 0
   879  002556  000000                    PASS:   0
   880  002560  000000                    FAST:   0
   881
   882  002562  000103                    MX:     .ASCIZ "C"
   883
   884  002564  046111  042514  040507    MILTST: .ASCIZ "ILLEGAL TEST"<15><12>
   885  002572  020114  042524  052123
   886  002600  005015     000
   887
   888          002604                    .EVEN
   889
```

```
                                        ; DATA SW 10 = 1 TO HALT AT END OF TEST
                                        ;
002604  104406                  LOCKUP: CKSWR
002606  032777  002000  176402          BIT #SW10,@SWR
002614  001401                          BEQ 1$
002616  000000                          HALT
                                        ;
                                        ; DATA SW 12 = 1 TO LOCK SCOPE LOOP ON TEST OK OR NOT
                                        ;
002620  032777  010000  176370  1$:     BIT #SW12,@SWR          ;IS LOOP ON TEST SWITCH SET
002626  001403                          BEQ 2$                  ;IF NOT SET GO ON TO NEXT TEST
002630  062716  000002                  ADD #2,@SP              ;IF SET RETURN TO FIRSTTEST
002634  000207                          RTS PC
002636  042737  040100  012752  2$:     BIC #40100,UNITSEL      ;CLEAR UNIT USED BITS
002644  000207                          RTS PC
```

```
                        .SBTTL TEST 1 - RXCS TEST PART I / INTERRUPT TEST PART I

                        ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING ALL RXCS WRITABLE BITS TO A 0
                        ; ARE [NOT] WRITTEN TO A 1

                        ;THE PROGRAM WRITES THE RXCS = 0

                        ;NO INTERRUPTS SHOULD OCCUR

                        ;THE RXCS SHOULD REMAIN UNCHANGED = 40 (DONE)

                        ;THE RXDB SHOULD = 0

002646  005077  176334  T1:     CLR @ RXCS                      ; WRITING RXCS TO 0, ALSO WRITES RXDB = 0
002652  012700  000040          MOV #40, R0
002656  017701  176324          MOV @ RXCS, R1
002662  020001                  CMP R0, R1
002664  001401                  BEQ 1$

                        ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A

002666  104000                  ERROR                           ; RXCS NOT = 40 (DONE)

     ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
     ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
     ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

          ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
          ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
          ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
          ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
          ;
          ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
          ;ANALYZE THE FOLLOWING AREA/S:
     ;
     ;M7846 (UNIBUS INTERFACE)
     ;
     ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
     ;
     ;BUS D06                 STUCK HIGH                      E4
     ; ----------             CAN'T SEND DATA                 E41
     ;B TRANSFER REQUEST      STUCK HIGH                      E15
     ;RX INIT                 STUCK LOW                       E36,E11
     ;BUS D15                 STUCK LOW                       E14
     ;D00(1) H                STUCK HIGH                      E22
     ;RX RUN                  STUCK LOW                       E18
     ;BUS D00                 STUCK LOW                       E7
     ;
     ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
     ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
     ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

002670  104413          1$:     SUBSCOPE
```

```
                        ;//////////////////////////////////////////////////////////////////////////////
                        ; RXDB AT "DONE"
                        ;       7       6       -       -       3       2       1       0       /
                        ;                                                                       /
                        ;       SEL                                     WRITE   INIT    PAR     /
                        ;       DRIVE   DD                              PROTECT [DONE]          /
                        ;       RDY                                                     CRC     /
                        ;                                                       (N/A)           /
                        ;//////////////////////////////////////////////////////////////////////////////


002672  005000                    CLR R0
002674  017701  176310            MOV @ RXDB, R1
002700  020100                    CMP R1, R0
002702  001401                    BEQ 3$

                        ; (R0) = 0 ; (R1) = ACTUAL RXDB ; (R2) = N/A

002704  104000                    ERROR                           ; RXDB NOT = 0

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
;
;BUS D01 -> D03         STUCK HIGH                      E7
;LOAD                   STUCK LOW                       E17,E18
; ----------            NO SACK                         E12
; ----------            CLOCK STUCK HIGH/LOW            E12
;IN                     STUCK LOW/HIGH                  E21
;RX BUSY                STUCK HIGH                      E22
;OUT                    STUCK LOW                       E22
;BUS D00/D02/D03        STUCK HIGH                      E4
;BUS -> RXCS            STUCK LOW                       E40
;REG SELECT             STUCK HIGH                      E23
;B DONE                 STUCK HIGH                      E19
; ----------            'B2' INPUT STUCK HIGH           E1
;
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

1016  002706  104413           3$:     SUBSCOPE
```

```
                        ; INTERRUPT TEST PART I ; NO INTERRUPTS SHOULD OCCUR

002710  013702  001204         MOV KRXVEC, R2
002714  010246                 MOV     R2,-(SP)          ;SAVE INTERRUPT VECTOR FOR
                                                          ;ERROR REPORT
002716  012722  002756         MOV #4$, (R2)+            ; RX01 VECTOR ADDRESS
002722  012722  000340         MOV #PR7, (R2)+
002726  012602                 MOV     (SP)+,R2          ;RESTORE INTERRUPT VECTOR FOR
                                                          ;ERROR REPORT
002730  005046                 CLR -(SP)                 ; PDP PRIORITY <ON>
002732  012746  002740         MOV #2$,-(SP)
002736  000002                 RTI
002740  000240         2$:     NOP
002742  000240                 NOP
002744  012746  000340         MOV #PR7,-(SP)            ; RESET PDP PRIORITY <7> OFF
002750  012746  002760         MOV #5$,-(SP)
002754  000002                 RTI

                        ; RETURN TO HERE WITH PDP PRIORITY = 7 IF AN RX01 INTERRUPT


                        ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A

002756  104000         4$:     ERROR                     ; UNEXPECTED RX01 INTERRUPT

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
;
;INT ENB                 STUCK HIGH                      E36
;
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

1065  002760  000004         5$:     SCOPE
1066  002762  000137  004274         JMP CEXIT                       ;END OF TEST 1
```

```
                                        .SBTTL TEST 2 - INTERRUPT TEST PART II / VECTOR ADDRESS VERIFICATION

                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT WRITING THE RXCS INTERRUPT ENABLE BIT
                                ; (BIT 6) = 1 DOES INDEED WRITE IT TO A 1, THEREFORE BECAUSE DONE = 1
                                ; AN INTERRUPT SHOULD OCCUR (THE PDP PRIORITY IS 0 )

002766  013702  001204         T2:     MOV KRXVEC, R2
002772  010246                          MOV R2,-(SP)            ;SAVE INTERRUPT VECTOR FOR
                                                                ;ERROR REPORT
002774  012722  003044                  MOV #1$, (R2)+          ; RX01 VECTOR ADDRESS
003000  012722  000340                  MOV #PR7, (R2)+
003004  012602                          MOV (SP)+,R2            ;RESTORE INTERRUPT VECTOR FOR
                                                                ;ERROR REPORT
003006  005046                          CLR -(SP)               ; PDP PRIORITY <ON>
003010  012746  003016                  MOV #6$,-(SP)
003014  000002                          RTI
003016  052777  000100  176162  6$:     BIS #BIT6, @ RXCS       ; SET RX01 INTERRUPT ENABLE
003024  000240                          NOP

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
;
; ----------            SACK FLOP LOCKED CLEAR          E13,E31,E36
; ----------            BBSY FLOP CLOCK LOCKED HIGH     E13,E31,E36,E33
;BUS REQUEST            STUCK HIGH                      E31
;BUS INTR               STUCK HIGH                      E31,E35,E8,E38
; ----------            SACK FLOP CLOCK LOCKED HIGH     E31,E21,E9,E12
; ----------            SACK FLOP CLOCK LOCKED LOW      E31,E21
; ----------            BBSY FLOP CLOCK LOCKED LOW      E31,E33,E9
;BUS D02/D04/D05        STUCK HIGH                      E38
;BUS SACK               STUCK HIGH                      E32
;BG OUT                 STUCK HIGH                      E32,E25,E28
;BUS REQUEST            STUCK LOW                       E39
; ----------            JUMPER (N1) NOT IN               ----------
;BUS D07                STUCK HIGH                      E35,E8
; ----------            GRANT FLOP CAN'T BE PRESET      E25,E9,E28
;B SER DATA             STUCK LOW                       E9
; ----------            GRANT FLOP 'Q' OUTPUT           E28
;                          STUCK HIGH
;
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
```

```
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 1125  003026  000240                          NOP
 1126  003030  012746  000340                  MOV #PR7,-(SP)                  ; RESET PDP PRIORITY <7> OFF
 1127  003034  012746  003042                  MOV #7$,-(SP)
 1128  003040  000002                          RTI
 1129
 1130                                  ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
 1131
 1132  003042  104000                  7$:     ERROR                           ; NO RX01 INTERRUPT OCCURRED
 1133

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
                ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
                ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
                ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
                ;
                ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
                ;ANALYZE THE FOLLOWING AREA/S:
        ;
        ;M7846 (UNIBUS INTERFACE)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ; ----------            BBSY FLOP LOCKED IN 'RESET'     E40,E39,E31
        ; ----------            INT ENB FLOP CLOCK LOCKED HIGH  E40
        ; ----------            INT ENB FLOP CLOCK LOCKED LOW   E40,E37
        ;RX INIT                STUCK HIGH                      E36
        ;BUS REQUEST            STUCK HIGH                      E36,E39,E32
        ; ----------            LOCKED IN INTERRUPT STATE       E36,E39
        ;BG OUT                 STUCK HIGH                      E33
        ;INT ENB(1) H           STUCK LOW                       E37,E1,E4
        ; ----------            GRANT FLOP 'Q' OUTPUT           E28
        ;                          STUCK LOW
        ;TRANSFER REQUEST       STUCK LOW                       E4
        ;BUS INTR               STUCK LOW                       E38
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 1168
 1169                                  ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
 1170
 1171  003044  005737  003060          1$:     TST CATERR                      ;DID MORE THAN ONE INTERRUPT
 1172                                                                          ;OCCUR DUE TO BOTH 'DONE' & 'IE'
 1173                                                                          ;BITS SET BUT 'RQST INTR' FLOP
 1174                                                                          ;NEVER GETTING CLEARED?
 1175  003050  001004                          BNE     DEATH                   ;BRANCH IF YES
 1176  003052  105237  003060                  INCB    CATERR                  ;OTHERWISE, SET FLAG TO INDICATE
 1177                                                                          ;ONE INTERRUPT OCCURRED & GO ON
 1178  003056  000402                          BR      OK2GO
```

```
1179  003060  000000                  CATERR: .WORD   0                       ;THIS IS THE INTERRUPT INDICATOR.
1180                                                                          ;1ST TIME THRU THIS TEST THE IN-
1181                                                                          ;DICATOR SHOULD ALWAYS BE ZERO;
1182                                                                          ;IF NON-ZERO A MULTIPLE OF
1183                                                                          ;INTERRUPTS HAS OCCURRED DUE TO
1184                                                                          ;THE AFOREMENTIONED CONDITION
1185  003062  000000                  DEATH:  HALT                            ;NO SENSE GOING ON - YOU ARE
1186                                                                          ;LOCKED IN THE INTERRUPT STATE!
1187                                                                          ;PRESS THE 'CONT' SWITH (IF
1188                                                                          ;PRESENT) TO GO ON  AT YOUR OWN
1189                                                                          ;RISK
1190  003064  104413                  OK2GO:  SUBSCOPE                        ;EVERYTHING APPEARS TO BE OK
1191  003066  005037  003060                  CLR     CATERR                  ;CLEAR INTERRUPT INDICATOR FOR
1192                                                                          ;NEXT POSSIBLE PASS THRU THIS TEST
1193
1194                                  ; THE RXCS SHOULD = 140 (INTERRUPT ENABLE+DONE)
1195
1196  003072  012700  000140                  MOV #140, R0
1197  003076  020077  176104                  CMP R0, @ RXCS
1198  003102  001403                          BEQ 2$
1199  003104  017701  176076                  MOV @ RXCS, R1
1200
1201                                  ; (R0) = 140 ; (R1) = ACTUAL RXCS ; (R2) = N/A
1202
1203  003110  104000                          ERROR                           ; THE RXCS NOT = 140
1204

      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
      ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

              ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
              ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
              ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
              ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
              ;
              ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
              ;ANALYZE THE FOLLOWING AREA/S:
      ;
      ;M7846 (UNIBUS INTERFACE)
      ;
      ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
      ;
      ;BUS D07                 STUCK HIGH                      E4
      ;BUS D06                 STUCK HIGH                      E1
      ;
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
      ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

1229  003112  104413          2$:     SUBSCOPE
1230
1231
1232                                  ; TEST 2 - CONT'D
1233
1234                                  ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT BIT 6 OF THE RXCS (INTERRUPT ENABLE)
```

```
                                ; CAN BE CLEARED AFTER IT WAS KNOWN TO BE SET

003114  013702  001204                 MOV KRXVEC, R2
003120  010246                         MOV     R2,-(SP)                ;SAVE INTERRUPT VECTOR FOR
                                                                        ;ERROR REPORT
003122  012722  003212                 MOV #4$, (R2)+                  ; RX11 VECTOR ADDRESS
003126  012722  000340                 MOV #PR7, (R2)+
003132  012602                         MOV     (SP)+,R2                ;RESTORE INTERRUPT VECTOR FOR
                                                                        ;ERROR REPORT
003134  042777  000100  176044         BIC #BIT6, @ RXCS               ; CLEAR THE RX11 INTERRUPT ENABLE BIT

                                ; THE RXCS SHOULD = 40 (DONE)

003142  012700  000040                 MOV #40, R0
003146  020077  176034                 CMP R0, @ RXCS
003152  001403                         BEQ 3$
003154  017701  176026                 MOV @ RXCS, R1

                                ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A

003160  104000                         ERROR                           ; RXCS NOT = 40
003162  104413                 3$:     SUBSCOPE
                                ; NO RX11 INTERRUPTS SHOULD OCCUR [YET]

003164  005046                         CLR -(SP)                       ; PDP PRIORITY <ON>
003166  012746  003174                 MOV #10$,-(SP)
003172  000002                         RTI
003174  000240                 10$:    NOP
003176  000240                         NOP
003200  012746  000340                 MOV #PR7,-(SP)                  ; PDP PRIORLITY <OFF> 7
003204  012746  003214                 MOV #11$,-(SP)
003210  000002                         RTI

                                ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN UNEXPECTED RX11 INTERRUPT
                                ; WHILE CLEARING THE RX11 INTERRUPT ENABLE BIT 6

                                ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A

003212  104000                 4$:     ERROR                           ; UNEXPECTED RX11 INTERRUPT
003214  104413                 11$:    SUBSCOPE

                                ; AN RX11 INTERRUPT SHOULD OCCUR [NOW]

003216  013702  001204                 MOV KRXVEC, R2
003222  010246                         MOV     R2,-(SP)                ;SAVE INTERRUPT VECTOR FOR
                                                                        ;ERROR REPORT
003224  012722  003274                 MOV #5$, (R2)+                  ; RX11 VECTOR ADDRESS
003230  012722  000340                 MOV #PR7, (R2)+
003234  012602                         MOV     (SP)+,R2                ;RESTORE INTERRUPT VECTOR FOR
                                                                        ;ERROR REPORT
003236  005046                         CLR -(SP)                       ; PDP PRIORITY <ON>
003240  012746  003246                 MOV #12$,-(SP)
003244  000002                         RTI
003246  052777  000100  175732 12$:    BIS #BIT6, @ RXCS               ; SET RX11 INTERRUPT ENABLE BIT
003254  000240                         NOP
003256  000240                         NOP
```

```
1291  003260  012746  000340          MOV #PR7,-(SP)
1292  003264  012746  003272          MOV #13$,-(SP)
1293  003270  000002                  RTI
1294
1295                                ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
1296
1297  003272  104000                13$:    ERROR                           ; NO RX11 INTERRUPT OCCURRED
1298
1299                                ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT
1300
1301  003274  000004                5$:     SCOPE
1302  003276  042777  000100  175702        BIC #BIT6, @ RXCS               ; CLEAR THE RX11 INTERRUPT ENABLE
1303
1304  003304  000137  004274              JMP CEXIT                         ;END OF TEST 2
```

```
                                        .SBTTL TEST 3 - INTERRUPT TEST PART III / PRIORITY LEVEL VERIFICATION PART I

                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
                                ; THE PROGRAM SETS THE PDP PRIORITY TO 1 LESS THAN THE DEVICE LEVEL
                                ;(DEVICE LEVEL SPECIFIED BY CONTENTS OF LOCATION 'BRLEV:' -- NORMALLY 5)
                                ; AN RX01 INTERRUPT SHOULD OCCUR
                                ; IF NO INTERRUPT OCCURS THEN THE PRIORITY LEVEL OF THE RX11 IS [NOT] = NORMAL
                                ;DEVICE LEVEL OF 5 OR THE DEVICE LEVEL AS SPECIFIED BY THE CONTENTS OF
                                ;LOCATION 'BRLEV:' WHICH MAY HAVE BEEN CHANGED BY THE USER BEFORE PROGRAM
                                ; EXECUTION, BUT MAYBE SOME VALUE LESS THAN THE CONTENTS OF LOCATION 'BRLEV:'.
                                ; NOTE: IF THERE IS NO HARDWARE "PSW" THIS TEST WILL BE SKIPPED.

003310  005001          T3:     CLR     R1                      ;INDICATOR TO CPU PRIORITY
                                                                ;ROUTINE TO DROP CPU PRIORITY
                                                                ;TO 1 LESS THAN DEVICE LEVEL
003312  013702  001204          MOV KRXVEC, R2
003316  010246                  MOV     R2,-(SP)                ;SAVE INTERRUPT VECTOR FOR
                                                                ;ERROR REPORT
003320  012722  003432          MOV #1$, (R2)+                  ; RX01 VECTOR ADDRESS
003324  012722  000340          MOV #PR7, (R2)+
003330  012602                  MOV     (SP)+,R2                ;RESTORE INTERRUPT VECTOR FOR
                                                                ;ERROR REPORT
003332  013746  000004          MOV     4,-(SP)                 ;SAVE 'BUSERR' TIMEOUT 'PC'
003336  012737  003354  000004  MOV     #2$,4                   ;SET TIMEOUT VECTOR
003344  012737  000200  177776  MOV     #PR4,PSW                ;SET LEVEL TO 4 IF 'PSW' EXISTS
003352  000404                  BR      3$                      ;GO TO RESET VECTOR 4 & DO TEST
003354  022626          2$:     CMP     (SP)+,(SP)+             ;CORRECT STACK FROM BUS TIMEOUT
003356  012637  000004          MOV     (SP)+,4                 ;RESTORE TIMEOUT VECTOR TO 'BUSERR'
003362  000427                  BR      4$                      ;NO HARDWARE PSW - SKIP THIS TEST
003364  012637  000004  3$:     MOV     (SP)+,4                 ;RESET TIMEOUT VECTOR TO 'BUSERR'
003370  004737  006226          JSR     PC,CPUPRI               ;CALCULATE PRIORITY LEVEL OF CPU
                                                                ;BASED ON CURRENT DEVICE PRIORITY
                                                                ;LEVEL RESIDING IN LOC. 'BRLEV'
003374  010046                  MOV     R0,-(SP)        ;;PUT NEW PS ON STACK
003376  012746  003404          MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
003402  000002                  RTI                     ;;POP NEW PC AND PS
003404                  64$:
003404  052777  000100  175574  BIS #BIT6, @ RXCS                ; SET THE RX01 INTERRUPT ENABLE
003412  000240                  NOP
003414  000240                  NOP
003416  013746  000340          MOV     PR7,-(SP)       ;;PUT NEW PS ON STACK
003422  012746  003430          MOV     #65$,-(SP)      ;;PUT NEW PC ON  STACK
003426  000002                  RTI                     ;;POP NEW PC AND PS
003430                  65$:

                                ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A

003430  104000                  ERROR                           ;PRIORITY LEVEL IS NOT = CONTENTS
                                                                ;OF 'BRLEV:' (NORMALLY 5) BUT
                                                                ;MAYBE SOME VALUE LESS THAN THE
                                                                ;THE CURRENT CONTENTS OF 'BRLEV:'

                                ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT

003432  000004          1$:     SCOPE
003434  042777  000100  175544  BIC #BIT6, @ RXCS                ; CLEAR THE RX11 INTERRUPT ENABLE
```

```
                                4$:     JMP CEXIT                               ;END OF TEST 3

                                        .SBTTL TEST 4 - INTERRUPT TEST PART IV / PRIORITY VERIFICATION PART II

                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THE PRIORITY OF THE RX11 INTERRUPT REQUEST LINE
                                ; THE PROGRAM SETS THE PDP PRIORITY = THE DEVICE LEVEL, (NORMALLY 5 OR THE CONTENTS OF L
                                ;NO RX01 INTERRUPTS SHOULD OCCUR
                                ; IF AN INTERRUPT DOES OCCUR THEN THE PRIORITY LEVEL OF THE RX11 IS [NOT]
                                ; = THE NORMAL DEVICE LEVEL OF 5, OR WHATEVER IS THE VALUE IN LOCATION 'BRLEV:'
                                ; BUT MAYBE SOME VALUE GREATER THAN THE CONTENTS OF LOC. 'BRLEV:'
                                ; NOTE: IF THERE IS NO HARDWARE "PSW" THIS TEST WILL BE SKIPPED.

003446  005001                  T4:     CLR     R1                              ;INDICATOR TO CPU PRIORITY ROUTINE
                                                                                ;TO DROP CPU PRIORITY 1 LEVEL
                                                                                ;LESS THAN THE DEVICE LEVEL
003450  013702  001204                  MOV KRXVEC, R2
003454  010246                          MOV     R2,-(SP)                        ;SAVE INTERRUPT VECTOR FOR
                                                                                ;ERROR REPORT
003456  012722  003574                  MOV #1$, (R2)+                          ; RX01 VECTOR ADDRESS
003462  012722  000340                  MOV #PR7, (R2)+
003466  012602                          MOV     (SP)+,R2                        ;RESTORE INTERRUPT VECTOR FOR
                                                                                ;ERROR REPORT
003470  052701  000200                  BIS     #BIT7,R1                        ;SET INDICATOR TO CPU PRIORITY
                                                                                ;ROUTINE TO SET CPU PRIORITY LEVEL
                                                                                ;TO THE SAME LEVEL AS THE DEVICE
003474  013746  000004                  MOV     4,-(SP)                         ;SAVE 'BUSERR' TIMEOUT 'PC'
003500  012737  003516  000004          MOV     #3$,4                           ;SET TIMEOUT VECTOR
003506  012737  000240  177776          MOV     #PR5,PSW                        ;SET LEVEL TO 5 IF 'PSW' EXISTS
003514  000404                        BR      4$                              ;GO ON TO RESET VECTOR & DO TEST
003516  022626                  3$:     CMP     (SP)+,(SP)+                     ;CORRECT STACK FROM BUS TIMEOUT
003520  012637  000004                  MOV     (SP)+,4                         ;RESTORE TIMEOUT VECTOR TO 'BUSERR'
003524  000430                        BR      5$                              ;NO HARDWARE PSW - SKIP THIS TEST
003526  012637  000004          4$:     MOV     (SP)+,4                         ;RESET TIMEOUT VECTOR TO BUSERR
003532  004737  006226                  JSR     PC,CPUPRI                       ;CALCULATE CPU PRIORITY LEVEL TO
                                                                                ;BE THE SAME AS THE DEVICE LEVEL
                                                                                ;I.E. - SAME AS CONTENTS OF LOC.
                                                                                ;'BRLEV'
003536  010046                          MOV     R0,-(SP)        ;;PUT NEW PS ON STACK
003540  012746  003546                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
003544  000002                        RTI                     ;;POP NEW PC AND PS
003546                          64$:
003546  052777  000100  175432          BIS     #BIT6,@RXCS                     ;SET RX01 INTERRUPT ENABLE
003554  000240                        NOP
003556  000240                        NOP
003560  013746  000340                  MOV     PR7,-(SP)       ;;PUT NEW PS ON STACK
003564  012746  003572                  MOV     #65$,-(SP)      ;;PUT NEW PC ON  STACK
003570  000002                        RTI                     ;;POP NEW PC AND PS
003572                          65$:
003572  000401                        BR 2$

                                ; RETURN TO HERE WITH THE PDP PRIORITY = 7 IF AN RX01 INTERRUPT

                                ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A

003574  104000                  1$:     ERROR                                   ;PRIORITY LEVEL NOT = TO CONTENTS
                                                                                ;OF LOCATION 'BRLEV:' (NORMALLY 5)
```

```
1417
1418                                                    ;BUT MAYBE SOME VALUE GREATER THAN
1419                                                    ;THE CONTENTS SPECIFIED BY LOC.
1420  003576  000004                    2$:             ;'BRLEV:'
                                        SCOPE
1421  003600  042777  000100  175400                    BIC #BIT6, @ RXCS       ; CLEAR THE RX01 INTERRUPT ENABLE
1422  003606  000137  004274            5$:             JMP CEXIT               ;END OF TEST 4
```

```
                                .SBTTL TEST 5 - INIT [PROGRAMMED] / RST

                        ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT SETTING THE RXCS BIT 14
                        ; CAUSES AN RX01 PROGRAMMED SUBSYSTEM INITIALIZE

                        ; THE RXCS SHOULD = 40 (DONE)

                        ; THE RXDB SHOULD = 4, OR 104, OR 204, OR 304

003612  052777  040000  175366  T5:    BIS #BIT14, @ RXCS          ; RX01 PROGRAMMED INITIALIZE
003620  004737  006622          1$:    JSR PC, SDN                 ; WAIT FOR THE DONE BIT
003624  000775                         BR 1$
003626  012700  000040                 MOV #40, R0                 ; PROGRAM EXPECTS RXCS = 40 (DONE)
003632  017701  175350                 MOV @ RXCS, R1              ; ACTUAL RXCS
003636  020100                         CMP R1, R0
003640  001401                         BEQ 2$

                        ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A

003642  104000                         ERROR                       ; RXCS NOT = 40

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
                ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
                ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
                ;MODULE M7727.
        ;
        ;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
        ;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
        ;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
        ;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
        ;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
        ;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
        ;
        ;M7727 (READ/WRITE CONTROL)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ; ----------            'J' INPUT LOCKED LOW            E16
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

003644  104413          2$:    SUBSCOPE

                        ;////////////////////////////////////////////////////////////////////////////////
                        ;
                        ;       7       6       -       -       3       2       1       0               /
                        ;
                        ;       SEL                             WRITE   INIT    PAR                     /
                        ;       DRIVE   DD                      PROTECT [DONE]          CRC             /
```

```
                                        ;       RDY                                                   (N/A)              /
                                        ;
                                        ;//////////////////////////////////////////////////////////////////////////////

                                        ; THE RXDB SHOULD = 4, 104, IF TESTING UNIT 1
                                        ; OR 204, OR 304 IF TESTING UNIT 0 (SEL. DRIVE RDY. BIT SET)

 003646  012700  000204                         MOV #204,R0
 003652  017702  175332                         MOV @RXDB,R2
 003656  010201                                 MOV R2,R1
 003660  042701  000100                         BIC #BIT6,R1                    ;CLEAR DELETED DATA BIT
 003664  105737  012752                         TSTB UNITSEL                    ;WAS UNIT 0 SELECTED
 003670  100404                                 BMI 3$
 003672  042701  000200                         BIC #BIT7,R1                    ;UNIT 0 WAS NOT SELECTED
 003676  042700  000200                         BIC #BIT7,R0                    ;CLEAR UNIT 0 RDY BITS
 003702  020100                         3$:     CMP R1, R0
 003704  001401                                 BEQ 4$

                                        ; (R0) = 4, OR 204
                                        ; (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
                                        ; (R2) = ACTUAL RXDB

 003706  104000                                 ERROR                           ; RXDB NOT = 4, OR 104, OR 204, OR 304


 ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
 ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
 ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

         ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
         ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
         ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
         ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
         ;
         ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
         ;ANALYZE THE FOLLOWING AREA/S:
 ;
 ;M7846 (UNIBUS INTERFACE)
 ;
 ;       SIGNAL NAME                REASON                       POSSIBLE CHIPS
 ;
 ;RX INIT                 STUCK HIGH                        E11,E36
 ;IN                      STUCK HIGH                        E11
 ;BUS D05                 STUCK LOW                         E1
 ;RX DATA                 STUCK LOW                         E14
 ; ----------             SACK FLOP CLOCK LOCKED LOW        E9
 ;B SER DATA              STUCK LOW                         E9
 ;DONE                    STUCK HIGH                        E22
 ; ----------             LOAD PULSE STUCK LOW              E3
 ;
 ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
 ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
 ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 003710  104413                         4$:     SUBSCOPE
```

```
                                        ; TEST 5 CONT'D - RXCS TEST PART II / RST

                                        ; THE PURPOSE OF THIS TEST IS TO VERIFY THE RST COMMAND #12+GO
                                        ; AND THAT THE DONE BIT IS CLEARED AFTER THE FUNCTION IS ISSUED.

003712  012777  000013  175266                  MOV #13, @ RXCS                 ; RST COMMAND
003720  032777  000040  175260                  BIT #BIT5, @ RXCS               ; TEST DONE BIT (SHOULD = 0)
003726  001404                                  BEQ 5$
003730  017701  175252                          MOV @ RXCS, R1
003734  005000                                  CLR R0

                                        ; (R0) = 0 ; (R1) = ACTUAL RXCS ; (R2) = N/A

003736  104000                                  ERROR                           ; DONE BIT NOT = 0
003740  104413                          5$:     SUBSCOPE
003742  004737  006622                  9$:     JSR PC, SDN                     ; WAIT FOR DONE FLAG
003746  000775                                  BR 9$
003750  012700  000040                          MOV #40, R0                     ; PROGRAM EXPECTS RXCS = 40 (DONE)
003754  017701  175226                          MOV @ RXCS, R1                  ; ACTUAL RXCS
003760  020100                                  CMP R1, R0
003762  001401                                  BEQ 10$

                                        ; (R0) = 40 ; (R1) = ACTUAL RXCS ; (R2) = N/A

003764  104000                                  ERROR                           ; RXCS NOT = 40

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
                ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
                ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
                ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
                ;
                ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
                ;ANALYZE THE FOLLOWING AREA/S:
        ;
        ;M7846 (UNIBUS INTERFACE)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ;
        ; ----------          PARITY FLOP CLOCK LOCKED HIGH     E2
        ; ----------          PARITY FLOP 'Q' OUTPUT            E2
        ;                        LOCKED HIGH
        ;RX DATA              STUCK LOW                         E6

        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

003766  104413                          10$:    SUBSCOPE

                                        ; THE RXDB SHOULD = 200 (IF DRIVE 0 IS READY), OR 0 IF UNIT 0 IS NOT SELECTED
                                        ; MAYBE 300 (IF DRIVE 0 IS READY AND SECTOR 1 WAS WRITTEN WITH DELETED DATA)
```

```
1590
1591  003770  017702  175214          MOV @ RXDB, R2                  ; ACTUAL RXDB
1592  003774  010201                  MOV R2, R1
1593  003776  042701  000100          BIC #BIT6, R1                   ; CLEAR N/A DELETED DATA BIT
1594  004002  012700  000200          MOV #200,R0                     ;EXPECT UNIT 0 RDY SET
1595  004006  105737  012752          TSTB UNITSEL
1596  004012  100403                  BMI 11$
1597  004014  042701  000200          BIC #BIT7,R1                    ; UNIT 0 NOT SELECTED
1598  004020  005000                  CLR R0                          ; DISREGUARD RDY BITS
1599  004022  020100          11$:    CMP R1, R0
1600  004024  001401                  BEQ 12$
1601
1602                          ; (R0) = 0 OR 200
1603                          ; (R1) = ACTUAL RXDB MINUS DELETED DATA BIT#6
1604                          ; (R2) = ACTUAL RXDB
1605
1606  004026  104000                  ERROR                           ; RXDB NOT = 200, OR NOT = 0
1607  004030  000004          12$:    SCOPE
1608  004032  000137  004274          JMP CEXIT                       ;END OF TEST 5
```

```
                                        .SBTTL TEST 6 - FILL BUFFER TRANSFER LENGTH VERIFICATION

                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THE TRANSFER LENGTH OF THE FUNCTION
                                ; "FILL BUFFER" OF THE RX01 MICROCONTROLLER

                                ; 128 BYTE TRANSFERS SHOULD OCCUR

004036  012737  000006  012352  T6:     MOV #6, PAT                     ; COUNT PATTERN
004044  004737  004054          T6FILL  JSR PC, T6FILL
004050  000137  004274          T6END   JMP CEXIT                       ;END OF TEST 6

                                ; EXECUTE THE FOLLOWING " MOV #1, @ RXCS " INSTEAD OF " INC @ RXCS " FOR LOOPS

004054  005002                  T6FILL: CLR R2
004056  012777  000001  175122          MOV #1, @ RXCS                  ;FILL BUFFER FUNCTION [0 + GO BIT]
004064  004737  012306                  JSR PC, GETPATTERN
004070  012704  017456                  MOV #BUFADR, R4
004074  004737  004112          1$:     JSR PC, FBEB                    ; SUBROUTINE TO FILL/EMPTY BUFFER
004100  000207                          RTS PC                          ; EXIT SUBROUTINE T6FILL

004102  112477  175102                  MOVB (R4)+, @ RXDB              ; FILL THE SECTOR BUFFER
004106  005202                          INC R2                          ;INC R2 FOR BYTE COUNT
004110  000771                          BR 1$

                                ; SUBROUTINE TO FILL AND EMPTY THE SECTOR BUFFER

004112  004737  006606          FBEB:   JSR PC, STR                     ; TEST FOR TRANSFER REQUEST
004116  000403                          BR 1$
004120  062716  000002                  ADD #2, @ SP                    ; ADJUST FOR EXIT FROM THIS SUBROUTINE
004124  000207                          RTS PC                          ; EXIT TO SERVICE TRANSFER REQUEST

004126  004737  006622          1$:     JSR PC, SDN
004132  000767                          BR FBEB                         ; NOT TRANSFER REQUEST OR DONE
004134  005777  175046                  TST @ RXCS                      ;TEST FOR ERROR FLAG
004140  100003                          BPL 3$
004142  017701  175042                  MOV @RXDB,R1

                                ; (R0) = N/A ; (R1) = RXDB (STATUS A) ; (R2) = ACTUAL # OF TRANSFERS

004146  104000                          ERROR                           ; UNEXPECTED RX11 ERROR FLAG

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
```

```
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ; ----------                    PARITY FLOP CAN'T BE CLEARED    E2
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 1673  004150  104413                  3$:     SUBSCOPE
 1674
 1675                                  ; 128 BYTES SHOULD HAVE BEEN TRANSFERRED TO OR FROM THE SECTOR BUFFER
 1676
 1677  004152  022702  000200                  CMP #200,R2
 1678  004156  001401                          BEQ 2$
 1679
 1680                                  ; (R0) = N/A ; (R1) = N/A ; (R2) = ACTUAL # OF TRANSFERS
 1681
 1682  004160  104000                          ERROR                           ; INCORRECT TRANSFER LENGTH
 1683

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
                ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
                ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
                ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
                ;
                ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
                ;ANALYZE THE FOLLOWING AREA/S:
        ;
        ;M7846 (UNIBUS INTERFACE)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ;RX DATA                 STUCK LOW                       E19,E6
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 1707  004162  000004                  2$:     SCOPE
 1708
 1709  004164  000207                          RTS PC                          ; EXIT SUBROUTINE FBEB
```

```
1710
1711                                    .SBTTL TEST 10 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART II
1712
1713                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THAT THE PREVIOUS EMPTY BUFFER TEST
1714                                ; DID NOT EMPTY AND DESTROY THE CONTENTS OF THE SECTOR BUFFER
1715                                ;NOTE: TEST 6 MUST BE RUN BEFORE THIS TEST.
1716  004166  000240                T10:    NOP                             ; NOP FOR T10 " FAST " DEFINITION
1717
1718                                    .SBTTL TEST 7 - EMPTY BUFFER XFER LENGTH AND CONTENT VERIFICATION PART I
1719
1720                                ; THE PURPOSE OF THIS TEST IS TO VERIFY THE TRANSFER LENGTH OF THE FOUNCTION
1721                                ; "EMPTY BUFFER" AND ALSO TO VERIFY THE CONTENTS OF THE SECTOR BUFFER
1722                                ; NOTE TEST 6 MUST BE RUN BEFORE THIS TEST
1723
1724  004170  004737  004200        T7:     JSR PC, T7EMPTY
1725  004174  000137  004274                JMP CEXIT                       ;END OF TEST 7 OR 10
1726  004200  005002                T7EMPTY: CLR R2
1727  004202  012777  000003  174776        MOV #3, @ RXCS                  ; ISSUE THE COMAND TO EMPTY BUFFER
1728  004210  012704  017456                MOV #BUFADR, R4
1729  004214  004737  004112        2$:     JSR PC, FBEB                    ; SUBROUTINE TO EMPTY BUFFER
1730  004220  000207                        RTS PC                          ; EXIT SUBROUTINE T7EMPTY
1731
1732                                ; EMPTY BUFFER AND VERIFY THE CONTENTS
1733
1734  004222  112400                        MOVB (R4)+, R0                  ; EXPECTED CONTENTS OF SECTUR BUFFER
1735  004224  117701  174760                MOVB @ RXDB, R1                 ; ACTUAL CONTENTS OF SECTOR BUFFER
1736  004230  005202                        INC R2                          ; ACTUAL # TRANSFERS TO THIS ERROR
1737  004232  020001                        CMP R0, R1
1738  004234  001401                        BEQ 1$
1739
1740                                ; IF AN ERROR OCCURS:
1741
1742                                ; (R0) - EXPECTED CONTENTS OF SECTOR BUFFER
1743                                ; (R1) - ACTUAL CONTENTS FROM SECTOR BUFFER
1744                                ; (R2) - TOTAL # OF ACTUAL TRANSFERS
1745
1746  004236  104000                        ERROR                           ; DATA " TO " SB NOT = DATA " FROM "
1747

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
;
;RX RUN                  STUCK LOW                       E24
```

```
;LOAD              STUCK LOW                    E18
; ----------                BINARY COUNTER 'RESET'          E34
;                              LOCKED HIGH
;RX DATA                    STUCK HIGH                      E3
;BUS D04/D05                STUCK HIGH/LOW                  E1,E4
;BUS D02                    STUCK HIGH                      E7
;

        ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
        ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
        ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
        ;MODULE M7727.
;
;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
;
;M7727 (READ/WRITE CONTROL)
;
;       SIGNAL NAME                REASON                     POSSIBLE CHIPS
;
; ----------            REPLACE E5                          ----------
;
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

004240  000004                          1$:     SCOPE
004242  000764                                  BR 2$

                                                .SBTTL TEST 12 - FILL/EMPTY BUFFER ALL 1'S

                                        ; PROGRAMMING NOTE: THE FOLLOWING " MOV " INSTEAD OF " INC " FOR LOOPS

004244  012737  000001  012352          T12:    MOV #1, PAT
004252  000402                                  BR .+6

                                                .SBTTL TEST 11 - FILL/EMPTY BUFFER ALL 0'S

004254  005037  012352                  T11:    CLR PAT
004260  004737  004054                          JSR PC, T6FILL
004264  004737  004200                          JSR PC, T7EMPTY
004270  000137  004274                          JMP CEXIT                       ;END OF TEST 11 OR 12
```

```
1812  004274  012737  000103  002562  CEXIT:  MOV #'C,MX                       ;EOP INDICATOR FOR END OF CONTROL TESTS
1813  004302  000137  002200                  JMP MORETESTS
1814
1815  004306  012737  000104  002562  DEXIT:  MOV #'D,MX                       ;EOP INDICATOR FOR END OF DRIVE TESTS
1816  004314  000137  002200                  JMP MORETESTS
1817
1818                                          .SBTTL TEST 13 DRIVE READY VERIFICATION
1819
1820                                  ;THIS TEST VERIFIES THAT DRIVE READY WILL SET FOR ALL SELECTED DRIVES.
1821
1822                                  ;THIS SECTION OF TEST FOR UNIT 0 DRY NO A READ STATUS A FUNCTION
1823
1824  004320  105737  012752          T13:    TSTB UNITSEL                    ;WAS UNIT 0 SELECTED
1825  004324  100020                          BPL 2$                          ;NO, GO TEST UNIT 1
1826  004326  012777  000013  174652          MOV #13,@RXCS                   ;READ STATUS A UNIT 0
1827  004334  004737  006622          1$:     JSR PC,SDN                      ;WAIT FOR DONE FLAG
1828  004340  000775                          BR 1$
1829  004342  012700  000200                  MOV #200,R0                     ;EXPECT DRIVE READY TO BE SET
1830  004346  017702  174636                  MOV @RXDB,R2                    ;ACTUAL RXDB
1831  004352  010201                          MOV R2,R1
1832  004354  042701  000100                  BIC #BIT6,R1                    ;CLEAR DD BIT IF SET
1833  004360  020001                          CMP R0,R1                       ;WAS " DRY " SET AND INITDONE CLEARED
1834  004362  001401                          BEQ 2$
1835
1836                                  ;R0 = 200 ; R1 = RXDB MINUS DD BIT ;R2 = ACTUAL RXDB
1837
1838  004364  104000                          ERROR
1839  004366  104413          2$:     SUBSCOPE
1840
1841                                  ;TEST FOR UNIT 1 DRIVE READY ON A READ STATUS A FUNCTION
1842
1843  004370  005737  012752                  TST UNITSEL                     ;WAS UNIT 1 SELECTED
1844  004374  100020                          BPL 3$                          ;NO, GO TO END OF TEST
1845  004376  012777  000033  174602          MOV #33,@RXCS                   ;READ STATUS A FOR DRIVE 1
1846  004404  004737  006622          4$:     JSR PC,SDN                      ;WAIT FOR DONE FLAG
1847  004410  000775                          BR 4$
1848  004412  012700  000200                  MOV #200,R0                     ;EXPECT " DRY " TO BE SET
1849  004416  017702  174566                  MOV @RXDB,R2
1850  004422  010201                          MOV R2,R1
1851  004424  042701  000100                  BIC #BIT6,R1                    ;CLEAR DD BIT IF ANY
1852  004430  020100                          CMP R1,R0                       ;IS DRY SET AND INITDONE CLEARED
1853  004432  001401                          BEQ 3$
1854
1855                                  ;R0 = 200 ; R1 = RXDB MINUS DD BIT ; R2 = ACTUAL RXDB
1856
1857  004434  104000                          ERROR
1858

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
                ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
                ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
                ;MODULE M7727.
```

```
        ;
        ;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
        ;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
        ;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
        ;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
        ;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
        ;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
        ;
        ;M7727 (READ/WRITE CONTROL)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;


        ;       NOTE:  MAKE SURE THE DRIVES ARE CONNECTED CORECTLY,THE
        ;               DISKETTES INSERTED,AND THE DOORS OF THE SELECTED DRIVES
        ;               ARE CLOSED. IF THE THESE CONDITIONS ARE NOT SET THERE
        ;               WILL BE AN ERROR AT THIS POINT.

        ;SEL DK1                STUCK LOW                       E13,E14
        ;DK1 INDX               STUCK HIGH/LOW                  E15
        ;SEL INDX               STUCK HIGH/LOW                  E15
        ; ----------            'A2' INPUT STUCK LOW            E15
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

1896  004436  000004                 3$:     SCOPE
1897  004440  000137  004306                 JMP DEXIT                       ;END OF TEST 13
```

```
1898                                    .SBTTL TEST 14 - ERROR FLAG AND B-CODE VERIFICATION PART I
1899
1900                                    ;THE PURPOSE OF THIS TEST IS TO VERIFY THAT TRYING TO READ A NON-EXISTANT
1901                                    ;SECTOR WILL CAUSE AN ERROR, AND THE CORRECT ERROR CODE WILL BE PUT
1902                                    ;INTO THE RXDB WHEN THE B STATUS IS READ.
1903
1904                                    ;THIS SECTION INSURES THAT ONLY 2 TR FLAGS WERE REQUIRED TO TAKE THE
1905                                    ;DISKETTE ADDRESS, AND THAT AN ERROR DOES EXIST.
1906
1907  004444  005002                    T14:    CLR R2
1908  004446  005000                            CLR R0
1909  004450  105737  012752                     TSTB UNITSEL                    ;IS UNIT 0 SELECTED
1910  004454  100004                            BPL 10$
1911  004456  012777  000007  174522            MOV #7,@RXCS                    ;SET READ SECTOR FUNCTION AND GO
1912  004464  000403                            BR 11$
1913  004466  012777  000027  174512  10$:     MOV #27,@RXCS                   ;SEND READ FUNCTION TO UNIT 1
1914  004474  004737  005134          11$:     JSR PC,ILLADR                   ;SEND THE BAD SECTOR ADDRESS
1915  004500  000406                            BR 1$                           ;PREMATURE ERROR CONDITION
1916  004502  012700  100040                    MOV #100040,R0                  ;EXPECT ERROR AND DONE BITS
1917  004506  017701  174474                    MOV @RXCS,R1                    ;SAVE THE RXCS
1918  004512  020001                            CMP R0,R1
1919  004514  001401                            BEQ 2$
1920
1921                                    ;R0 = 100040 ; R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS
1922
1923  004516  104000                    1$:     ERROR
1924

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
                ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
                ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
                ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
                ;
                ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
                ;ANALYZE THE FOLLOWING AREA/S:
        ;
        ;M7846 (UNIBUS INTERFACE)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ;BUS D15                 STUCK HIGH                      E14,E9,E40,E24
        ;RX RUN                  STUCK LOW                       E41
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

1949  004520  104413                    2$:     SUBSCOPE
1950
1951                                    ;T14 CONT. - THIS SECTION TESTS THAT NO PARITY OR CRC ERROR OCCURRED
1952                                    ;ON PREVIOUS FUNCTION.
1953
```

```
1954  004522  005000                        CLR R0                          ;STATUS A SHOULD BE CLEAR
1955  004524  017702  174460                MOV @RXDB,R2
1956  004530  010201                        MOV R2,R1
1957  004532  042701  000100                BIC #BIT6,R1
1958  004536  020001                        CMP R0,R1
1959  004540  001401                        BEQ 3$
1960
1961                                  ;R0 = 0 ; R1 = RXDB MINUS DD BIT ; R2 = ACTUAL RXDB
1962
1963  004542  104000                        ERROR
1964  004544  104413                  3$:   SUBSCOPE
1965
1966                                  ;T14 CONT. - THIS SECTION TESTS THAT NO ERROR CONDITIONS EXIST ON A
1967                                  ;READ STATUS B FUNCTION.
1968
1969  004546  012777  000017  174432        MOV #17,@RXCS                   ;SET READ STATUS B FUNCTION
1970  004554  004737  006622          4$:   JSR PC,SDN                      ;WAIT FOR DONE FLAG
1971  004560  000775                        BR 4$
1972  004562  005777  174420                TST @RXCS                       ;IS THE ERROR BIT SET
1973  004566  100007                        BPL 5$                          ;NO, GO ON TO NEXT SECTION
1974  004570  012700  000040                MOV #40,R0                      ;YES,SET UP FOR ERROR
1975  004574  017701  174406                MOV @RXCS,R1
1976  004600  017702  174404                MOV @RXDB,R2
1977
1978                                  ;R0 = 40 ; R1 = RXCS ; R2 = RXDB
1979
1980  004604  104000                        ERROR                           ;RXCS NOT = 40
1981  004606  104413                  5$:   SUBSCOPE
1982
1983                                  ;T14 CONT. - THIS SECTION TESTS THE B-CODE FOR " CAN'T FIND SECTOR " #70
1984
1985  004610  012700  000070                MOV #70,R0
1986  004614  017701  174370                MOV @RXDB,R1
1987  004620  005002                        CLR R2
1988  004622  020001                        CMP R0,R1                       ;IS THE B-CODE = 70
1989  004624  001401                        BEQ 6$
1990
1991                                  ;R0 = 70 ; R1 = ACTUAL B-CODE ; R2 = N/A
1992
1993  004626  104000                        ERROR                           ;RXDB NOT = 70
1994

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
                ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
                ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
                ;MODULE M7727.
        ;
        ;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
        ;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
        ;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
        ;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
        ;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
```

```
;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
;
;M7727 (READ/WRITE CONTROL)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
;
;INIT                    STUCK HIGH                      E16
;SEL TRK 0               STUCK HIGH                      E15
;DK1 TRK 0               STUCK HIGH                      E15
;SEL DK0                 STUCK LOW                       E15,E14,E13
;WT GATE                 STUCK HIGH                      E5
;SEL WT PROT             STUCK LOW                       E5
;RAW DATA                STUCK HIGH/LOW                  E14
;STEP                    STUCK LOW                       E14
; ----------             REPLACE E4                       ----------
;
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

2030  004630  000004                   6$:     SCOPE
2031  004632  000137  004306                   JMP DEXIT                       ;END OF TEST 14
2032
```

```
                                .SBTTL TEST 15 - ERROR FLAG AND B-CODE VERIFICATION PART II

                        ;THIS TEST VERIFIES THAT TRYING TO WRITE,USING DELETED DATA MODE, ON A
                        ;NON-EXISTANT SECTOR WILL PRODUCE AN ERROR AND THE CORRECT B-CODE IS PRODUCED
                        ;THIS SECTION SENDS OUT AN ILLEGAL SECTOR ADDRESS AND EXPECTS AN ERROR
                        ; NOTE TEST 14 MUST BE RUN BEFORE THIS TEST

004636  005000          T15:    CLR R0
004640  005002                  CLR R2
004642  105737  012752          TSTB UNITSEL                    ;WAS UNIT 0 SELECTED
004646  100004                  BPL 10$
004650  012777  000015  174330  MOV #15,@RXCS                   ;SET WRITE DELETED DATA FUNCTION
004656  000403                  BR 11$
004660  012777  000035  174320  10$:    MOV #35,@RXCS           ;SEND WTR DD FUNCTION TO UNIT 1
004666  004737  005134          11$:    JSR PC,ILLADR           ;SEND THE ILLEGAL SECTOR ADDRESS
004672  000406                  BR 1$                           ;PREMATURE ERROR CONDITION
004674  012700  100040          MOV #100040,R0                  ;EXPECT ERROR AND DONE FLAGS
004700  017701  174302          MOV @RXCS,R1
004704  020001                  CMP R0,R1
004706  001401                  BEQ 2$

                        ;R0 = 100040 ;R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS

004710  104000          1$:     ERROR                           ;RXCS NOT = 100040
004712  104413          2$:     SUBSCOPE

                        ;T15 CONT. - THIS SECTION TESTS THAT THERE IS NO PARITY, CRC ERROR
                        ;AND THAT THE DELETED DATA BIT IS SET.

004714  005002                  CLR R2
004716  012700  000100          MOV #100,R0                     ;EXPECT DELETED DATA BIT TO BE SET
004722  017701  174262          MOV @RXDB,R1
004726  020001                  CMP R0,R1
004730  001401                  BEQ 3$

                        ; R0 = 100 ; R1 = ACTUAL RXDB ; R2 = N/A
004732  104000                  ERROR                           ;DELETED DATA NOT SET OR OTHER ERRORS
004734  104413          3$:     SUBSCOPE

                        ;T15 CONT. - THIS SECTION TESTS FOR THE B-CODE FOR ILLEGAL SECTOR.

004736  012777  000017  174242  MOV #17,@RXCS                   ;SET READ STATUS B FUNCTION
004744  004737  006622          4$:     JSR PC,SDN              ;WAIT FOR DONE FLAG
004750  000775                  BR 4$
004752  012700  000070          MOV #70,R0
004756  017701  174226          MOV @RXDB,R1
004762  020001                  CMP R0,R1
004764  001401                  BEQ 5$

                        ; R0 = 70 ; R1 = ACTUAL B-CODE ; R2 = N/A
004766  104000                  ERROR                           ;RXDB NOT = 70
004770  000004          5$:     SCOPE
004772  000137  004306          JMP DEXIT                       ;END OF TEST 15
```

```
                                        .SBTTL TEST 16 - ERROR FLAG AND B-CODE VERIFICATION PART III

                                ;THIS TEST VERIFIES THAT A WRITE FUNCTION TO A NON-EXISTANT SECTOR WILL
                                ;PRODUCE AN ERROR AND A B-CODE OF 70. THE DELETED DATA BIT SHOULD ALSO BE CLEARED
                                ;THIS SECTION TRANSFERS AN ILLEGAL SECTOR ADDRESS FOR A WRITE FUNCTION
                                ; NOTE TEST 14 MUST BE RUN BEFORE THIS TEST

004776  005000          T16:    CLR R0
005000  005002                  CLR R2
005002  105737  012752          TSTB UNITSEL                    ;WAS UNIT 0 SELECTED
005006  100004                  BPL 10$
005010  012777  000005  174170  MOV #5,@RXCS                    ;SET THE WRITE FUNCTION
005016  000403                  BR 11$
005020  012777  000025  174160  10$:  MOV #25,@RXCS             ;SEND WRITE FUNCTION TO UNIT 1
005026  004737  005134          11$:  JSR PC,ILLADR              ;SEND THE ILLEGAL ADDRESS
005032  000406                  BR 1$                           ;PREMATURE ERROR CONDITION
005034  012700  100040          MOV #100040,R0                  ;EXPECT ERROR AND DONE BITS SET
005040  017701  174142          MOV @RXCS,R1
005044  020001                  CMP R0,R1
005046  001401                  BEQ 2$

                                ; R0 = 100040 ; R1 = ACTUAL RXCS ; R2 = # OF TR FLAGS

005050  104000          1$:     ERROR
005052  104413          2$:     SUBSCOPE

                                ;T16 CONT. - TESTS FOR NO PARITY, CRC ERRORS, AND NO DELETED DATA BIT

005054  005002                  CLR R2
005056  005000                  CLR R0                          ;NO BITS SHOULD BE SET IN THE RXDB
005060  017701  174124          MOV @RXDB,R1
005064  020001                  CMP R0,R1
005066  001401                  BEQ 3$

                                ; R0 = 0 ; R1 = ACTUAL RXDB ; R2 = N/A
005070  104000                  ERROR                           ;SOME BIT IS SET IN THE RXDB
005072  104413          3$:     SUBSCOPE

                                ;T16 CONT. - TEST FOR CORRECT B-CODE FOR ILLEGAL SECTOR ADDRESS

005074  012777  000017  174104  MOV #17,@RXCS                   ;SET READ STATUS B FUNCTION
005102  004737  006622          4$:  JSR PC,SDN                 ;WAIT FOR DONE FLAG
005106  000775                  BR 4$
005110  012700  000070          MOV #70,R0
005114  017701  174070          MOV @RXDB,R1
005120  020001                  CMP R0,R1                       ;IS B-CODE = 70
005122  001401                  BEQ 5$                          ;YES, CONTINUE

                                ; R0 = 70 ; R1 = ACTUAL RXDB ; R2 = N/A
005124  104000                  ERROR
005126  000004          5$:     SCOPE
005130  000137  004306          JMP DEXIT                       ;END OF TEST 16
```

```
                                ;GENERATE AN ILLEGAL SECTOR ADDRESS
005134  004737  006606  ILLADR: JSR PC,STR               ;LOOK FOR A TR FLAG
005140  000402                  BR 2$                   ;NO TR FLAG, IS DONE SET
005142  005202                  INC R2                  ;TR FLAG COUNTER
005144  000404                  BR 3$
005146  004737  006622  2$:     JSR PC,SDN              ;LOOK FOR DONE FLAG
005152  000770                  BR ILLADR               ;NOT DONE RECHECK TR FLAG
005154  000430                  BR 1$                   ;DONE IS SET TOO EARLY GO TO ERROR
005156  005077  174026  3$:     CLR @RXDB               ;0 SECTOR ADDRESS (ILLEGAL)
005162  004737  006606  7$:     JSR PC,STR              ;LOOK FOR SECOND TR FLAG
005166  000402                  BR 5$                   ;NOT TR, IS IT DONE
005170  005202                  INC R2
005172  000404                  BR 6$                   ;TR FLAG SEND TRACK ADDRESS
005174  004737  006622  5$:     JSR PC,SDN              ;LOOK FOR DONE FLAG
005200  000770                  BR 7$                   ;NOT DONE, RECHECK TR FLAG
005202  000415                  BR 1$                   ;DONE TOO SOON GO TO ERROR
005204  005077  174000  6$:     CLR @RXDB               ;SEND TRACK ADDRESS OF 0
005210  004737  006606  11$:    JSR PC,STR              ;ARE THERE ANY MORE TR FLAGS
005214  000402                  BR 10$                  ;NO, LOOK FOR DONE
005216  005202                  INC R2                  ;YES
005220  000406                  BR 1$                   ;TOO MANY TR FLAGS OR MICROCONTROLLER
                                                        ;DID NOT DETECT THE ERROR
005222  004737  006622  10$:    JSR PC,SDN              ;LOOK FOR DONE FLAG
005226  000770                  BR 11$                  ;NOT DONE RETEST TR FLAG
005230  062716  000002          ADD #2,@SP
005234  000207          4$:     RTS PC
005236  017701  173744  1$:     MOV @RXCS,R1
005242  000774                  BR 4$
```

```
                                        .SBTTL TEST 17 - ILLEGAL TRACK ERROR VERIFICATION

                                ;THIS TEST VERIFIES THAT IF A TRACK ADDRESS LARGER THAN 114 (OCTAL) IS
                                ;ACCESSED, AN ERROR CONDITION WILL EXIST, AND A B-CODE WILL = 40

005244  005002          T17:    CLR R2
005246  005000                  CLR R0
005250  012777  000007  173730  MOV #7,@RXCS                    ;SET READ FUNCTION
005256  004737  006606          3$:     JSR PC,STR              ;LOOK FOR TR FLAG
005262  000401                  BR 1$                           ;NO TR FLAG CHECK DONE
005264  000410                  BR 2$
005266  004737  006622          1$:     JSR PC,SDN
005272  000771                  BR 3$
005274  017701  173706          MOV @RXCS,R1                    ;DONE OCCURRED TOO SOON SET UP FOR ERROR
005300  017702  173704          MOV @RXDB,R2
005304  000433                  BR 4$
005306  012777  000001  173674  2$:     MOV #1,@RXDB            ;SEND LEGAL SECTOR ADDRESS
005314  004737  006606          5$:     JSR PC,STR              ;LOOK FOR TR FLAG
005320  000401                  BR 6$
005322  000410                  BR 7$
005324  004737  006622          6$:     JSR PC,SDN
005330  000771                  BR 5$
005332  017701  173650          MOV @RXCS,R1                    ;DONE SET TOO EARLY
005336  017702  173646          MOV @RXDB,R2
005342  000414                  BR 4$
005344  012777  000115  173636  7$:     MOV #115,@RXDB          ;SEND ILLEGAL TRACK ADDRESS
005352  004737  006622          10$:    JSR PC,SDN              ;WAIT FOR DONE ON THE ERROR
005356  000775                  BR 10$
005360  012700  100040          MOV #100040,R0                  ;EXPECT ERROR AND DONE SET
005364  017701  173616          MOV @RXCS,R1
005370  020001                  CMP R0,R1
005372  001401                  BEQ 11$

                                ;TWO ERROR CONDITIONS TO REPORT
                                ;IF R0 = 0 THEN R1 = RXCS ;R2 = RXDB ON A DONE TOO SOON ERROR
                                ;IF R0 = 100040 THEN R1 = ACTUAL RXCS ; R2 = N/A

005374  104000          4$:     ERROR                           ;DONE SET TOO SOON OR NO ERROR OCCURRED

;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

        ;THE FOLLOWING IS A PRESENTATION OF POSSIBLE REASONS AS TO WHY
        ;THIS ERROR REPORT WAS GENERATED.  THE INFORMATION SHOWN IS
        ;BASED ON FAULT INSERTION RESULTS,  AND SHOULD PROVIDE LOGICAL
        ;AREAS TO CHECK FOR THE RELEVANT FAULT/S.
        ;
        ;IF THIS ERROR REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS
        ;ANALYZE THE FOLLOWING AREA/S:
;
;M7846 (UNIBUS INTERFACE)
;
;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
```

```
                                        ;
                                        ; ----------             INT ENB FLOP CLOCK LOCKED HIGH  E40
                                        ;
                                        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
                                        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
                                        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
005376  104413                          11$:    SUBSCOPE

                                        ;T17 CONT. - TEST THAT THERE WERE NO OTHER ERRORS THAN THE ILLEGAL TRACK ERROR EXPECTED
                                        ;AND THAT THE DELETED DATA BIT WAS CLEARED BY TEST 16.

005400  005000                                  CLR R0                  ;DD BIT CLEARED IN TEST 16 SO RXDB = 0
005402  005002                                  CLR R2
005404  017701  173600                          MOV @RXDB,R1
005410  020001                                  CMP R0,R1
005412  001401                                  BEQ 12$

                                        ; R0 = 0 ; R1 = ACTUAL RXDB ; R2 = N/A

005414  104000                                  ERROR
005416  104413                          12$:    SUBSCOPE

                                        ;T17 CONT. - VERIFIES THAT READ STATUS B HAS NO ERRORS

005420  012777  000017  173560                  MOV #17,@RXCS           ;SET READ STATUS B FUNCTION
005426  004737  006622                  13$:    JSR PC,SDN              ;WAIT FOR DONE FLAG
005432  000775                                  BR 13$
005434  005777  173546                          TST @RXCS               ;WAS THERE AN ERROR ON THIS FUNCTION
005440  100007                                  BPL 14$                 ;NO
005442  012700  000040                          MOV #40,R0
005446  017701  173534                          MOV @RXCS,R1
005452  017702  173532                          MOV @RXDB,R2

                                        ; R0 = 40 (DONE) ; R1 = ACTUAL RXCS ; R2 = ACTUAL RXDB

005456  104000                                  ERROR                   ;THERE WAS AN ERROR NO READ STATUS B
005460  104413                          14$:    SUBSCOPE
005462  005002                                  CLR R2
005464  012700  000040                          MOV #40,R0              ;B-CODE FOR ILLEGAL TRACK ADDRESS
005470  017701  173514                          MOV @RXDB,R1
005474  020001                                  CMP R0,R1
005476  001401                                  BEQ 15$                 ;B-CODE IS CORRECT

                                        ; R0 = 40 ; R1 = ACTUAL B-CODE ; R2 = N/A

005500  104000                                  ERROR
005502  000004                          15$:    SCOPE
005504  000137  004306                          JMP DEXIT               ;END OF TEST 17
```

```
2273
2274                                                   .SBTTL TEST 20 - SEEK VERIFICATION VIA READ FUNCTION
2275
2276                                           ; THE PURPOSE OF THIS TEST IS TO DO A READ FUNCTION  ON ALL
2277                                           ;SECTORS OF VARIOUS TRACKS ON THE DISKETTE. THIS WILL TEST FOR SEEK ERRORS
2278                                           ;FOR THOSE TRACK POSITIONS. UNLESS OTHERWISE SELECTED (IN OD/ID) THE TRACKS
2279                                           ;ACCESSED ARE 0 (OD), 52, 53 (BOTH SIDES OF THE WRITE CURRENT CHANGE), AND 114 (ID).
2280
2281
2282    005510                                 T20:
2283

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;
                ;IF THE DIAGNOSTIC GIVES AN ERROR REPORT WITH A FORMAT OF
                ;'TEST PC=' WHERE THE 'PC' IS WITHIN THE RANGE OF THIS TEST
                ;THEN THE POSSIBLE CHIPS VERSUS THE 'B' CODE (INTERPRETIVE
                ;ERROR CODE) PRINTED ARE AS FOLLOWS:
                ;
        ;       IF 'B' CODE WAS         POSSIBLE CHIPS
        ;
        ;
        ;               120             E5,E6
        ;               150             E13,E14,E16,E17
        ;               200             E5,E6
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

2305    005510  013702  001204                          MOV KRXVEC,R2                   ;SET UP INTERRUPT ADDRESSES
2306    005514  012722  011554                          MOV #INTSERV,(R2)+
2307    005520  012722  000340                          MOV #PR7,(R2)+
2308    005524  004737  007010                          JSR PC,RDONLY                   ;DO THE READ FUNCTION
2309    005530  000137  004306                          JMP DEXIT                       ;END OF TEST 20
2310
2311
2312                                                    .SBTTL TEST 21 - WRITE TEST
2313
2314                                           ;THE PURPOSE OF THIS TEST IS TO WRITE ALL ONES ON SECTOR 1 TRACK 1,
2315                                           ;AND VERIFY THAT THE DATA IN THE SECTOR BUFFER IS NOT MODIFIED.
2316
2317    005534  012737  000001  013154  T21:    MOV #1,TARGET
2318    005542  012737  000001  013444          MOV #1,TSECTOR
2319    005550  004737  012662                  JSR PC,GETUNIT
2320    005554  012737  000001  012352          MOV #1,PAT
2321    005562  004737  012306                  JSR PC,GETPATTERN
2322    005566  004737  011014                  JSR PC,ADJSUM                   ;SET CHECK SUM VALUES
2323    005572  005002                          CLR R2
2324    005574  012777  000001  173404          MOV #1,@RXCS                    ;SET FILL BUFFER FUNCTION
2325    005602  004737  004112          1$:     JSR PC,FBEB
2326    005606  000404                          BR 2$
2327    005610  112077  173374                  MOVB (R0)+,@RXDB
2328    005614  005202                          INC R2
```

```
 2329  005616  000771                        BR 1$
 2330  005620  012737  000005  007730  2$:   MOV #5,FUNCTION          ;SET WRITE FUNCTION
 2331  005626  004737  005636                JSR PC,T21X             ;GO ISSUE THE COMMAND
 2332  005632  000137  004306                JMP DEXIT               ;END OF TEST 21
```

```
2333  005636  004737  007662          T21X:   JSR PC,COMMWORD
2334  005642  004737  006622          3$:     JSR PC,SDN
2335  005646  000775                          BR 3$
2336  005650  005777  173332                  TST @RXCS                       ;IS THE ERROR BIT SET
2337  005654  100013                          BPL 4$                          ;NO ERRORS
2338  005656  017701  173326                  MOV @RXDB,R1                    ;SAVE STATUS A
2339  005662  012777  000017  173316          MOV #17,@RXCS                   ;SET READ STATUS B FUNCTION
2340  005670  004737  006622          5$:     JSR PC,SDN
2341  005674  000775                          BR 5$
2342  005676  017702  173306                  MOV @RXDB,R2                    ;SAVE STATUS B
2343
2344                                  ;R0 = ADDR OF LAST DATA BYTE ; R1 = STATUS A ; R2 = STATUS B
2345
2346  005702  104000                          ERROR
2347

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:

                ;IF THE FAULT CANNOT BE FOUND ON THE UNIBUS INTERFACE MODULE
                ;AND/OR THE FAULT IS NOT INHERENT TO THE UNIBUS INTERFACE MODULE
                ;M7846 THERE IS A POSSIBILITY OF ITS EXISTENCE ON THE READ/WRITE
                ;MODULE M7727.
        ;
        ;       NOTE:   ONLY APPROX. 30% OF THIS MODULE LENT ITSELF CONDUCIVE TO
        ;               THE FAULT INSERTION PROCESS; ERGO, THE RESOLUTION FOR FAULT
        ;               ANALYSIS OBTAINABLE BY THIS MODULE IS NOT VERY HIGH.
        ;               HOWEVER, ANALYSIS OF THE FOLLOWING AREA/S, IF THIS ERROR
        ;               REPORT WAS THE 1ST GIVEN IN A SERIES OF ERRORS, SHOULD AT
        ;               LEAST PLACE YOU WITHIN THE RELEVANT AREA ON THE MODULE.
        ;
        ;M7727 (READ/WRITE CONTROL)
        ;
        ;       SIGNAL NAME             REASON                  POSSIBLE CHIPS
        ;
        ;WT DATA                 STUCK LOW/HIGH                  E5
        ;WT GATE                 STUCK LOW                       E13
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

2376  005704  104413          4$:     SUBSCOPE
2377  005706  005000                  CLR R0
2378  005710  005001                  CLR R1
2379  005712  004737  004200          JSR PC,T7EMPTY                  ;EMPTY BUFFER AND CHECK CONTENTS
2380  005716  000207                  RTS PC
```

```
2381                                    .SBTTL TEST 22 - INITIALIZE IMPLIED READ
2382                                ;AFTER PREVIOUSLY WRITING A PATTERN ON SECTOR 1 TRACK 1, THIS TEST
2383                                ;CHANGES THE CONTENTS OF THE SECTOR BUFFER AND DOES A PROGRAMMED INITIALIZE.
2384                                ;AFTER WHICH THE SECTOR BUFFER MUST AGAIN CONTAIN THE DATA PREVIOUSLY
2385                                ;WRITTEN ON THAT SECTOR AND TRACK.
2386                                ;NOTE:  THIS TEST WILL ONLY WORK IF UNIT 0 IS SELECTED AND ON LINE.
2387
2388
2389  005720  105737  012752        T22:    TSTB UNITSEL                    ;IF UNIT 0 IS NOT SELECTED SKIP THIS TEST
2390  005724  100022                        BPL 2$
2391  005726  005037  012352                CLR PAT
2392  005732  004737  004054                JSR PC,T6FILL                   ;LOAD THE SECTOR BUFFER WITH 0
2393  005736  005237  012352                INC PAT                         ;RELOAD CORE BUFFER WITH 1'S
2394  005742  004737  012306                JSR PC,GETPATTERN
2395  005746  004737  011014                JSR PC,ADJSUM
2396  005752  052777  040001  173226        BIS #RECAL,@RXCS                ;SET THE INIT. BIT
2397  005760  004737  006622          1$:   JSR PC,SDN
2398  005764  000775                        BR 1$
2399  005766  004737  004200                JSR PC,T7EMPTY                  ;EMPTY THE SECTOR BUFFER AND CHECK IT.
2400  005772  000137  004306          2$:   JMP DEXIT                       ;END OF TEST 22
2401
2402
2403                                    .SBTTL TEST 23 - READ TEST
2404
2405                                ;THIS TEST VERIFIES THAT A READ FUNCTION DOES IN FACT LOAD THE SECTOR
2406                                ;BUFFER WITH DATA READ FROM THE SELECTED ADDRESS.
2407
2408  005776  005037  012352        T23:    CLR PAT
2409  006002  004737  004054                JSR PC,T6FILL                   ;LOAD SECTOR BUFFER WITH 0'S
2410  006006  005237  012352                INC PAT
2411  006012  004737  012662                JSR PC,GETUNIT
2412  006016  004737  012306                JSR PC,GETPATTERN               ;RELOAD CORE BUFFER WITH 1'S
2413  006022  004737  011014                JSR PC,ADJSUM                   ;SET UP FOR CHECK SUM
2414  006026  012737  000007  007730        MOV #7,FUNCTION                 ;SET READ FUNCTION AND GO
2415  006034  004737  005636                JSR PC,T21X                     ;ISSUE COMMAND,WAIT FOR DONE,& TEST DATA
2416  006040  000137  004306                JMP DEXIT                       ;END OF TEST 23
```

```
2417
2418                                    .SBTTL TEST 24 - DATA TRANSFER AND VERIFICATION
2419
2420                                ;THE PURPOSE OF THIS TEST IS TO WRITE THEN READ AND VERIFY DATA
2421                                ;ON ALL SECTORS OF THE SELECTED TRACKS. THE TEST ALTERNATES BETWEEN
2422                                ;DRIVES ON THE SELECTED TRACKS. DATA PATTERN IS A FLOATING 0.
2423
2424  006044  012737  000002  012352  T24:    MOV #2,PAT                      ;SET DATA PATTERN TO FLOATING 0
2425  006052  013702  001204          T24X:   MOV KRXVEC,R2                   ;SET INTERRUPT ADDRESSES
2426  006056  012722  011554                  MOV #INTSERV,(R2)+
2427  006062  012712  000340                  MOV #PR7,(R2)
2428  006066  004737  007040                  JSR PC,DRVSWP                   ;GO TRANSFER THE DATA
2429  006072  000137  004306                  JMP DEXIT                       ;END OF TEST 24 OR 25
2430
2431
2432                                    .SBTTL TEST 25 - DATA TRANSFER AND VERIFICATION VIA DELETED DATA MODE
2433
2434                                ;THIS TEST TRANSFERES DATA JUST LIKE TEST 24 EXCEPT IT USES THE
2435                                ;DELETED DATA FORMAT AND A DATA PATTERN OF FLOATING 1.
2436
2437  006076  012737  000003  012352  T25:    MOV #3,PAT                      ;SET DATA PATTERN TO FLOATING 1
2438  006104  000137  006052                  JMP T24X                        ;GO TRANSFER THE DATA
2439
2440                                    .SBTTL TEST 26 - HEAD "HOME" TEST
2441
2442                                ;THIS TEST MOVES THE HEAD OUT TO TRACK 12 (OCTAL) AND THEN WRITES/READ CHECKS
2443                                ;ALL SECTORS (RANDOM DATA) ON EACH TRACK. THE TRACK SEQUENCE
2444                                ;IS DECREMENT;>ED BACK TO TRACK 0 (HOME). AFTER COMPLETING
2445                                ;DRIVE 0 IT SWITCHES OVER TO DRIVE 1 DOING THE SAME TEST.
2446
2447
2448  006110  052737  000200  013164  T26:    BIS #BIT7,SEQUEN                ;SPECIAL DECREMENT SEQUENCE
2449  006116  012737  000007  012352          MOV #7,PAT                      ;SELECT RANDOM DATA
2450  006124  013702  001204                  MOV KRXVEC,R2
2451  006130  012722  011554                  MOV #INTSERV,(R2)+
2452  006134  012712  000340                  MOV #PR7,(R2)
2453  006140  004737  007114                  JSR PC,WTRDCK
2454  006144  042737  000200  013164          BIC #BIT7,SEQUEN
2455  006152  000137  004306                  JMP DEXIT                       ;END OF TEST 26
```

```
                                ;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
                                ;TO THE USER WHEN AN 'UNEXPECTED' BUS TIMEOUT TO LOCATION 4 OCCURS

006156  104401  017007        BUSERR: TYPE,   LOC4M                   ;TYPE MESSAGE INDICATING AN
                                                                      ;UNEXPECTED BUS TIMEOUT OCCURRED
006162  012646                        MOV     (SP)+,-(SP)     ;;SAVE (SP)+ FOR TYPEOUT
                                                                ;;SETUP TO TYPE PC WHERE TIMEOUT OCCURRED
006164  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
006166     006                      .BYTE   6                  ;;TYPE 6 DIGITS
006167     000                      .BYTE   0                  ;;SUPPRESS LEADING ZEROS
006170  104401  017122              TYPE,   PCM                     ;TYPE MESSAGE '=PC'
006174  012716  002504              MOV     #REBEGIN,(SP)           ;SET RETURN 'PC' TO START THE
                                                                      ;PROGRAM OVER AGAIN
006200  000002                      RTI                             ;RETURN TO BEGINNING OF PROGRAM

                                ;THE FOLLOWING SECTION OF CODE WILL ALLOW PROVIDING INFORMATION
                                ;TO THE USER WHEN AN 'UNEXPECTED' RESERVED INSTRUCTION TRAP TO LOCATION
                                ;10 OCCURS

006202  104401  017054        RESERR: TYPE,   LOC10M                  ;TYPE MESSAGE INDICATING AN
                                                                      ;UNEXPECTED RESERVED INSTRUCTION
                                                                      ;TRAP OCCURRED
006206  012646                        MOV     (SP)+,-(SP)     ;;SAVE (SP)+ FOR TYPEOUT
                                                                ;;SETUP TO TYPE PC WHERE RESERVED TRAP OCCURRED
006210  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
006212     006                      .BYTE   6                  ;;TYPE 6 DIGITS
006213     000                      .BYTE   0                  ;;SUPPRESS LEADING ZEROS
006214  104401  017122              TYPE,   PCM                     ;TYPE MESSAGE '=PC'
006220  012716  002504              MOV     #REBEGIN,(SP)           ;SET RETURN 'PC' TO START THE
                                                                      ;PROGRAM OVER AGAIN
006224  000002                      RTI                             ;RETURN TO BEGINNING OF PROGRAM

                                ;THIS ROUTINE WILL CALCULATE THE PRIORITY LEVEL FOR THE PROCESSOR
                                ;BASED ON THE CURRENT PRIORITY LEVEL OF THE DEVICE (CONTENTS OF 'BRLEV:')

006226  013700  001214        CPUPRI: MOV     BRLEV,R0                ;GET THE PROPOSED RX11 DEVICE
                                                                      ;INTERRUPT PRIORITY LEVEL VALUE
006232  105701                        TSTB    R1                      ;IS CPU LEVEL TO BE THE SAME AS
                                                                      ;THE DEVICE LEVEL OR 1 LESS?
006234  100401                        BMI     1$                      ;BRANCH IF SAME AS!
006236  005300                        DEC     R0                      ;DROP DEVICE LEVEL PRIORITY
                                                                      ;BY 1 LEVEL FOR PSW
006240  006300                1$:     ASL     R0                      ;FORM BITS <7-5> FOR PSW
006242  006300                        ASL     R0                      ;       .
006244  006300                        ASL     R0                      ;       .
006246  006300                        ASL     R0                      ;       .
006250  006300                        ASL     R0                      ;       .
006252  042700  000037              BIC     #37,R0                  ;ENSURE THAT T,N,Z,V, & C BITS
                                                                      ;FOR THE PROCESSOR ARE CLEAR
006256  000207                      RTS     PC                      ;RETURN TO MAINLINE CODE
```

```
                                                                    1  7
 2508                                    .SBTTL  " ERROR " TRAP SERVICE ROUTINE
 2509
 2510                                    ;;*****************************************************************
 2511                                    ;;*****************************************************************
 2512
 2513                                    ;                           " ERROR "
 2514
 2515                                    ;;*****************************************************************
 2516                                    ;;*****************************************************************
 2517
 2518  006260  011637  006522            XERROR: MOV @ SP, EPCSCOPE              ; RETURN ADDRESS FROM " ERROR"
 2519  006264  062737  000002  006522            ADD #2, EPCSCOPE                ; NOW (EPCSCOPE) = SUBSCOPE+2, OR SCOPE+2
 2520  006272  005237  006520            INCERRORS: INC ERRORS
 2521  006276  001775                            BEQ INCERRORS
 2522
 2523                                    ; DATA SW 13 = 0 TO PRINT APPROPRIATE ERROR MESSAGE
 2524
 2525  006300  104406                            CKSWR
 2526  006302  032777  020000  172706            BIT #SW13,@SWR
 2527  006310  001056                            BNE NOPRINT
 2528  006312  005037  002554                    CLR CCOUNT
 2529  006316  032737  000400  012752            BIT #BIT8,UNITSEL               ;WAS PREVIOUS ERROR REPORTED ON THIS PASS
 2530  006324  001002                            BNE 1$
 2531  006326  104401  015700                    TYPE, MXEHEADER
 2532
 2533  006332  104401  016162            1$:     TYPE, MCRLF
 2534  006336  011604                            MOV @ SP, R4                    ; ERADR
 2535  006340  162704  000002                    SUB #2, R4
 2536  006344  010446                            MOV R4, -(SP)
 2537  006346  104403                            TYPOS
 2538  006350     006                            .BYTE 6
 2539  006351     001                            .BYTE 1
 2540  006352  104401  016703                    TYPE, SPACE
 2541  006356  013746  002560                    MOV FAST, -(SP)                 ; FAST (FIRST ADDRESS OF SELECTED TEST)
 2542  006362  104404                            TYPON
 2543  006364  104401  016703                    TYPE, SPACE
 2544  006370  013746  006604                    MOV PCSCOPE, -(SP)              ; FAPT (FIRST ADDRESS OF PRESENT TEST)
 2545  006374  104404                            TYPON
 2546  006376  104401  016703                    TYPE, SPACE
 2547  006402  010246                            MOV R2, -(SP)                   ; BLANK
 2548  006404  104404                            TYPON
 2549  006406  104401  016703                    TYPE, SPACE
 2550  006412  010046                            MOV R0, -(SP)                   ; EXPECTED (GOOD) RESULT OF TEST
 2551  006414  104404                            TYPON
 2552  006416  104401  016703                    TYPE, SPACE
 2553  006422  010146                            MOV R1, -(SP)                   ; ACTUAL (BAD) RESULT OF TEST
 2554  006424  104404                            TYPON
 2555  006426  104401  016703                    TYPE,.SPACE
 2556  006432  013737  002556  006444            MOV PASS,2$
 2557  006440  004537  015642                    JSR R5,SGLDEC
 2558  006444  000000                    2$:     OPEN
```

```
                                        ; DATA SW 0 = 0 TO RING BELL AT ERROR

006446  052737  000400  012752  NOPRINT: BIS #BIT8,UNITSEL              ;SET HARD ERROR FLAG
006454  004737  006476                   JSR PC,DING

                                        ; DATA SW 15 = 1 TO HALT AT ERROR

006460  104406                  1$:     CKSWR
006462  032777  100000  172526          BIT #SW15,@SWR
006470  001401                          BEQ 2$
006472  000000                          HALT
006474  000002                  2$:     RTI

006476  104406                  DING:   CKSWR
006500  032777  000001  172510          BIT #SW0,@SWR
006506  001002                          BNE 1$
006510  104401  006516                  TYPE ,MABELL
006514  000207                  1$:     RTS PC

006516  000007                  MABELL: .ASCIZ <07>                      ; DING - A - LING
                                        .EVEN

006520  000000                  ERRORS: 0
006522  000000                  EPCSCOPE: 0
```

```
                                        .SBTTL " SCOPE " TRAP SERVICE ROUTINE

                                ;                       " SCOPE "

006524  005737  006520         XSCOPE: TST ERRORS
006530  001015                         BNE SCOPING

                                ; NO ERRORS HAVE BEEN DETECTED

                                ; JUST SET (PCSCOPE) = FIRST ADDRESS OF THE SCOPE LOOP

                                ; (IN CASE ERRORS ARE DETECTED LATER)

006532  005037  006520         NOSCOPE: CLR ERRORS
006536  011637  006604                 MOV @ SP, PCSCOPE
006542  000002                         RTI

                                ;                       " SUBSCOPE "

006544  005737  006520         XSUBSCOPE: TST ERRORS
006550  001001                         BNE 1$
006552  000002                         RTI                             ; NO ERRORS EXIST

                                ; ERRORS DO EXIST

                                ; IF THIS ERROR ADDRESS IS THE SAME ADDRESS WITHIN PROGRAM LOCATION " EPCSCOPE"

                                ; THEN THIS IS A SCOPING LOOP

                                ; IF NOT - THEN EXIT

006554  021637  006522         1$:     CMP @ SP, EPCSCOPE
006560  001401                         BEQ SCOPING
006562  000002                         RTI

                                ; SW 11 = 1 TO LOCK ON SCOPING LOOP

                                ; THIS IS A SCOPING LOOP

006564  104406                 SCOPING: CKSWR
006566  032777  004000  172422         BIT #SW11,@SWR
006574  001756                         BEQ NOSCOPE                     ;DO NOT LOOP ON ERROR
006576  013716  006604                 MOV PCSCOPE, @ SP
006602  000002                         RTI                             ; LOCK FOR SCOPE LOOP
006604  000000                 PCSCOPE:        0
```

```
                                        ; WAIT FOR TRANSFER REQUEST FLAG TO SET
                                        ;
006606  105777  172374          STR:    TSTB @ RXCS
006612  100002                         BPL RTSPC
006614  062716  000002                 ADD #2, @ SP                    ; ADJUST FOR EXIT
006620  000207                 RTSPC:  RTS PC
                                        ;
                                        ; WAIT FOR THE DONE FLAG TO SET
                                        ;
006622  032777  000040  172356  SDN:    BIT #BIT5, @ RXCS               ; TEST THE DONE BIT # 5
006630  001055                         BNE XSDN
006632  062737  000001  007004         ADD #1,HANGER
006640  005537  007006                 ADC HANGPL
006644  001401                         BEQ HUNGUP
006646  000207                         RTS PC
                                        ;
                                        ; THE DEVICE TEST IS HUNG - DONE HAS NOT SET
                                        ;
                                        ; (R0) = N/A ; (R1) = N/A ; (R2) = N/A
                                        ;
006650  104000                 HUNGUP: ERROR

        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
                ;
                ;       IF THE DIAGNOSTIC HITS THIS ERROR REPORT AND THE 'HUNG @ PC'
                ;       LOCATION IS WITHIN ONE OF THE TESTS TABULATED BELOW THEN THE
                ;       POSSIBLE CHIPS ARE AS FOLLOWS:
                ;
        ;       HUNG @ TEST             POSSIBLE CHIPS
        ;
        ; TEST 5 ( PART 1 )             E7,E34,E4,E22,E15,E1,E11,E37
        ; TEST 5 ( PART 2 )             E7,E34,E2,E14,E6,E1,E37,E8
        ; TEST 6                        E4,E15,E1,E19,E18,E22
        ;
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

                                        ;
006652  005037  007004                  CLR HANGER                      ;RESET THE HANG COUNTERS
006656  012737  177740  007006         MOV #177740,HANGPL
006664  104406                         CKSWR
006666  032777  020000  172322         BIT #SW13,@SWR
006674  001402                         BEQ 1$
006676  000137  002200                 JMP MORETESTS
006702  104401  006722          1$:    TYPE, MHUNGPC
                                        ;
                                        ; THE PC IS ALREADY ON THE STACK
                                        ;
006706  162716  000002                 SUB #2, @ SP
006712  104403                         TYPOS
006714     006                         .BYTE 6
006715     001                         .BYTE 1
006716  000137  002200                 JMP MORETESTS
```

```
 2684
 2685  006722  005015  042504  044526  MHUNGPC: .ASCIZ <15><12>''DEVICE TEST HUNG @ PC ''
 2686  006730  042503  052040  051505
 2687  006736  020124  052510  043516
 2688  006744  040040  050040  020103
 2689  006752     000
 2690  006753     040  040520  051523  MPASS:   .ASCIZ '' PASS ='' 
 2691  006760  036440     000
 2692          006764                           .EVEN
 2693
 2694  006764  005037  007004          XSDN:    CLR HANGER
 2695  006770  012737  177740  007006           MOV #177740,HANGPL
 2696  006776  062716  000002                   ADD #2, @ SP                    ; UPDATE FOR EXIT
 2697  007002  000207                           RTS PC
 2698  007004  000000                  HANGER: 0
 2699  007006  177740                  HANGPL: 177740
```

```
                                        .SBTTL DRIVE TEST SELECTION

                                        ;DO A READ ONLY FUNCTION ON ALL SECTORS.
                                        ;THIS DOES NOT VERIFY THE DATA, ONLY TESTS FOR CRC ERRORS.

007010  004737  013000        RDONLY:        JSR PC,INITTRACKS
007014  004737  012662                       JSR PC,GETUNIT
007020  004737  013132        1$:            JSR PC,GETTRACK
007024  004737  010042                       JSR PC,READ
007030  005337  013152                       DEC TRKCNTR
007034  001371                               BNE 1$
007036  000207                               RTS PC

                                        ;;*****************************************************************

                                        ;WRITE AND READ DATA ON SPECIFIED TRACK AND ALTERNATE
                                        ;DRIVES BEFORE GOING TO THE NEXT TRACK.

007040  004737  012306        DRVSWP:        JSR PC,GETPATTERN
007044  004737  013000                       JSR PC,INITTRACKS
007050  004737  012662        1$:            JSR PC,GETUNIT
007054  004737  013132                       JSR PC,GETTRACK
007060  004737  007162                       JSR PC,WRITE
007064  004737  010662                       JSR PC,READCHK
007070  004737  012662                       JSR PC,GETUNIT
007074  004737  007162                       JSR PC,WRITE
007100  004737  010662                       JSR PC,READCHK
007104  005337  013152                       DEC TRKCNTR
007110  001357                               BNE 1$
007112  000207                               RTS PC

                                        ;;*****************************************************************

                                        ;WRITE ALL SECTORS AND READ/VERIFY ALL SECTORS


007114  004737  012306        WTRDCK:        JSR PC,GETPATTERN
007120  004737  013000        XWTRDCK:       JSR PC,INITTRACKS
007124  004737  012662                       JSR PC,GETUNIT
007130  004737  013132        1$:            JSR PC,GETTRACK
007134  004737  007162                       JSR PC,WRITE
007140  004737  010662                       JSR PC,READCHK
007144  005337  013152                       DEC TRKCNTR
007150  001367                               BNE 1$
007152  004737  012754                       JSR PC,DONE             ;HAVE BOTH DRIVES BEEN TESTED
007156  000207                               RTS PC                  ;YES
007160  000757                               BR XWTRDCK              ;NO, GO TO OTHER UNIT
```

```
                                                .SBTTL  WRITE FUNCTION

007162  004737  013352          WRITE:          JSR PC,INITSECTOR       ;SET UP FIRST,LAST,AND SECTOR COUNTER
007166  004737  013450          XWRITE:         JSR PC,GETSECTOR        ;PICK UP NEXT SECTOR
007172  004737  011014          FILLBUF:        JSR PC,ADJSUM           ;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
007176  012746  007360                          MOV #FILLDONE,-(SP)     ;PUT GOOD RETURNON STACK
007202  012746  007250                          MOV #FILLER,-(SP)       ;PUT ERROR RETURN ON STACK
007206  005037  011500                          CLR BYTECNTR
007212  005046                          CLR     -(SP)                   ;LOWER 'CPU' LEVEL
007214  012746  007222                  MOV     #1$,-(SP)               ;SET RETURN 'PC'
007220  000002                          RTI                             ;GET 'CPU' LEVEL INTO 'PSW'
007222  012777  000101  171756  1$:             MOV #FB1E,@RXCS         ;EXECUTE FILLBUFER COMMAND
007230  105777  171752          FILLFLAG:       TSTB @RXCS              ;TEST FOR TRANSFER REQUEST FLAG
007234  100375                                  BPL FILLFLAG
007236  112077  171746          XFRBYTE:        MOVB (R0)+,@RXDB        ;TRANSFER DATA BYTE
007242  005237  011500                          INC BYTECNTR
007246  000770                                  BR FILLFLAG             ;WAIT FOR NEXT TR FLAG

007250  005726                  FILLER:         TST (SP)+               ;REMOVE THE DONE RETURN FROM THE STACK
007252  012737  016430  007314                  MOV #MFIL,PTYP1+2       ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYPOUT 1
007260  012737  007166  007356                  MOV #XWRITE,PCONT+2     ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
007266  012737  007172  007352                  MOV #FILLBUF,PLOOP+2    ;IF LOOP ON ERROR RETURN THROUGH PLOOP
007274  000137  007300                          JMP PARTEST             ;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY

007300  104406                  PARTEST:        CKSWR
007302  032777  020000  171706                  BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
007310  001006                                  BNE CONT4
007312  104401  000000          PTYP1:          TYPE ,OPEN              ;PRINT THE PARITY ERROR MESSAGE
007316  104401  016641                          TYPE ,MPAR
007322  104401  016162                          TYPE ,MCRLF
007326  104406                  CONT4:          CKSWR
007330  005777  171662                          TST @SWR                ;TEST HALT ON ERROR SWITCH
007334  100001                                  BPL CONT13
007336  000000                  HLT6:           HALT                    ;HALT ON ERROR
007340  032777  004000  171650  CONT13:         BIT #SW11,@SWR          ;TEST LOOP ON ERROR SWITCH
007346  001402                                  BEQ PCONT               ;IF NOT SET GO TO NEXT SECTOR
007350  000137  007172          PLOOP:          JMP FILLBUF             ;RETURN TO LOOP ON TEST THROUGH HERE
007354  000137  010160          PCONT:          JMP NEXTRD              ;GO TO NEXT SECTOR THROUGH HERE

007360  005037  007004          FILLDONE:       CLR HANGER
007364  012746  007456                          MOV #WRTDONE,-(SP)      ;SET GOOD RETURN ON STACK
007370  012746  007472                          MOV #WRTER,-(SP)        ;SET ERROR RETURN ON STACK
007374  112737  000105  007730                  MOVB #WRTIE,FUNCTION    ;SET FUNCTION WORD TO WRITE
007402  022737  006076  006604                  CMP #T25,PCSCOPE        ;IS THIS THE DELETED DATA TEST
007410  001003                                  BNE 1$
007412  112737  000115  007730                  MOVB #WTDDIE,FUNCTION
007420  004737  007662          1$:             JSR PC,COMMWORD         ;TRANSFER COMMAND TO DRIVE
007424  005046                          CLR     -(SP)                   ;LOWER 'CPU' LEVEL
007426  012746  007434                  MOV     #2$,-(SP)               ;SET RETURN 'PC'
007432  000002                          RTI                             ;GET 'CPU' LEVEL INTO 'PSW'
007434  032777  000040  171544  2$:             BIT #DONEBIT,@RXCS      ;WAIT FOR DONE
007442  001774                                  BEQ 2$
007444  005237  007004          3$:             INC HANGER              ;WAIT FOR INTERRUPT
007450  001375                                  BNE 3$
007452  000137  011510                          JMP NOINTER             ;NO INTERRUPT ERROR
```

```
2802  007456  005337  013442          WRTDONE:        DEC SECCNTR             ;TEST SECTOR COUNTER
2803  007462  001001                                  BNE 2$                  ;NOT LAST SECTOR GO TO NEXT ONE
2804  007464  000207                                  RTS PC
2805  007466  000137  007166          2$:             JMP XWRITE
2806
2807  007472  005726                  WRTER:          TST (SP)+               ;REMOVE THE DONE RETURN FROM THE STACK
2808  007474  032737  000002  012174                  BIT #BIT1,ASTAT         ;IS THIS A PARITY ERROR
2809  007502  001413                                  BEQ WRTSEK              ;NO, IT MUST BE A SEEK ERROR
2810                                ;PARITY ERROR DURING A WRITE FUNCTION
2811  007504  012737  016632  007314                  MOV #MWRITE,PTYP1+2     ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYPOUT 1
2812  007512  012737  007456  007356                  MOV #WRTDONE,PCONT+2    ;IF NO LOOP GO TO NEXT SECTOR
2813  007520  012737  007360  007352                  MOV #FILLDONE,PLOOP+2   ;IF LOOP RETURN THROUGH PLOOP TO REWRITE
2814  007526  000137  007300                          JMP PARTEST             ;GO INC LOG AND TEST FOR RETRY
2815
2816                              ;SEEK ERROR DURING A WRITE FUNCTION
2817  007532  012737  007172  007632  WRTSEK:         MOV #FILLBUF,SEKRTY+2   ;SETUP FOR WRT RETRY ON SEEK ERROR
2818                                                                          ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1,
2819                                                                          ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
2820                                                                          ;TO REWRITE THE CORRECT DATA THE PROGRAM
2821                                                                          ;MUST REFILL THE SECTOR BUFFER.
2822  007540  012737  016632  007570                  MOV #MWRITE,STYP1+2     ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYPEOUT 1
2823  007546  004737  007554                          JSR PC,SEEKER           ;RECORD SEEK ERROR
2824  007552  000741                                  BR WRTDONE              ;GO TO NEXT SECTOR CAN'T FIND THIS ONE
2825
2826  007554  104406                  SEEKER:         CKSWR
2827  007556  032777  020000  171432                  BIT #SW13,@SWR          ;CHECK DON'T PRINT ERROR SWITCH
2828  007564  001004                                  BNE SWHLT1
2829  007566  104401  016632          STYP1:          TYPE ,MWRITE            ;PRINT WRITE (READ) SEEK ERROR
2830  007572  004737  007634                          JSR PC,SEKTYP
2831  007576  104406                  SWHLT1:         CKSWR
2832  007600  005777  171412                          TST @SWR                ;TEST THE HALT ON ERROR SWITCH
2833  007604  100001                                  BPL CONT14
2834  007606  000000                  HLT7:           HALT                    ;HALT ON THE ERROR
2835  007610  004737  007734          CONT14:         JSR PC,HOME             ;RECALIBRATE ON SEEK ERRORS
2836  007614  104406                                  CKSWR
2837  007616  032777  004000  171372                  BIT #SW11,@SWR          ;CHECK THE LOOP ON ERROR SWITCH
2838  007624  001001                                  BNE SEKRTY              ;IF SET LOOP ON THE ERROR THROUGH SEEK RETRY.
2839  007626  000207                                  RTS PC
2840  007630  000137  007172          SEKRTY:         JMP FILLBUF             ;RETRY WRITE COMMAND (READ COMAND)
2841
2842  007634  104401  016617          SEKTYP:         TYPE ,MSEEK             ;TYPE SEEK ERROR
2843  007640  104401  016074                          TYPE ,MPRES             ;TYPE ADDRESS OF TRACK MOVED FROM
2844  007644  013746  013156                  MOV     PRESTRK,-(SP)   ;;SAVE PRESTRK FOR TYPEOUT
2845  007650  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
2846  007652     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
2847  007653     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
2848  007654  104401  016162                          TYPE ,MCRLF
2849  007660  000207                                  RTS PC
2850
```

```
2851  007662  153737  012752  007730  COMMWORD:   BISB UNITSEL,FUNCTION   ;SET UNIT SELECTION BIT IN COMMAND WORD
2852  007670  013777  007730  171310              MOV FUNCTION,@RXCS       ;SEND OUT COMMAND TO DRIVE
2853  007676  004737  006606          1$:         JSR PC,STR               ;WAIT FOR TR FLAG
2854  007702  000775                              BR 1$
2855  007704  113777  013444  171276              MOVB TSECTOR,@RXDB       ;SEND OUT TARGET SECTOR
2856  007712  004737  006606          2$:         JSR PC,STR               ;WAIT FOR TR FLAG
2857  007716  000775                              BR 2$
2858  007720  113777  013154  171262              MOVB TARGET,@RXDB        ;SEND OUT TARGET TRACK
2859  007726  000207                              RTS PC
2860
2861  007730  000000          FUNCTION:   0
2862  007732  000000          DATACK:     0                                ;DATA CHECK ON CRC ERROR FLAG
2863
2864  007734  104406          HOME:       CKSWR
2865  007736  032777  000400  171252              BIT #SW8,@SWR            ;TEST NO RECAL SWITCH
2866  007744  001035                              BNE RTN
2867  007746  012777  040001  171232              MOV #RECAL,@RXCS         ;ISSUE RECAL FUNCTION
2868  007754  004737  006622          2$:         JSR PC,SDN
2869  007760  000775                              BR 2$
2870  007762  005777  171220                      TST @RXCS                ;WAS THERE AN ERROR
2871  007766  100021                              BPL XHOME                ;NO
2872  007770  104406                              CKSWR
2873  007772  032777  020000  171216              BIT #BIT13,@SWR          ;YES, SHOULD IT BE PRINTED
2874  010000  001002                              BNE 1$                   ;NO
2875  010002  004737  012200                      JSR  PC,RDCODE
2876  010006  104406          1$:         CKSWR
2877  010010  005777  171202                      TST @SWR                 ;TEST HALT ON ERROR SWITCH
2878  010014  100001                              BPL 3$
2879  010016  000000                              HALT
2880  010020  032777  004000  171170  3$:         BIT #SW11,@SWR           ;TEST LOOP ON ERROR SWITCH
2881  010026  001342                              BNE HOME
2882  010030  000207                              RTS PC
2883  010032  012737  000001  013156  XHOME:     MOV #1,PRESTRK           ;SET THE PRESENT TRACK TO TRACK 1
2884  010040  000207          RTN:        RTS PC
2885
```

```
                                        .SBTTL  READ DATA FROM THE DISKETTE

010042  004737  013352          READ:   JSR PC,INITSECTOR
010046  004737  013450          XREAD:  JSR PC,GETSECTOR
010052  005037  007732          REREAD: CLR DATACK              ;CLEAR CRC DATA CHECK FLAG
010056  005037  007004                  CLR HANGER
010062  012746  010136                  MOV #RDDONE,-(SP)       ;SET GOOD RETURN ON STACK
010066  012746  010170                  MOV #RDERR,-(SP)        ;SET READ ERROR RETURN ON STACK
010072  112737  000107  007730          MOVB #RDIE,FUNCTION
010100  004737  007662                  JSR PC,COMMWORD
010104  005046                  CLR     -(SP)                   ;LOWER 'CPU' LEVEL
010106  012746  010114          MOV     #1$,-(SP)               ;SET RETURN 'PC'
010112  000002                  RTI                             ;GET 'CPU' LEVEL INTO 'PSW'
010114  032777  000040  171064  1$:     BIT #DONEBIT,@RXCS      ;WAIT FOR DONE BIT
010122  001774                  BEQ 1$
010124  005237  007004          2$:     INC HANGER              ;WAIT FOR INTERRUPT
010130  001375                  BNE 2$
010132  000137  011510                  JMP NOINTER             ;NO INTERRUPT ON DONE

010136  022737  005510  006604  RDDONE: CMP #T20,PCSCOPE        ;IS THIS THE READ ONLY TEST (T20)
010144  001405                  BEQ NEXTRD              ;YES,DON'T CHECK FOR DELETED DATA
010146  004737  010436                  JSR PC,DDCHK            ;CHECK FOR DELETED DATA INDICATOR
010152  005701                  TST R1                  ;BIT 15 OF R1 IS READ 1 SECTOR FLAG
010154  100001                  BPL NEXTRD
010156  000207                  RTS PC                  ;IF SET,GO VERIFY DATA JUST READ
010160  005337  013442          NEXTRD: DEC SECCNTR
010164  001330                  BNE XREAD
010166  000207                  RTS PC                  ;READ FUNCTION IS DONE

010170  005726                  RDERR:  TST (SP)+               ;REMOVE THE DONE RETURN FROM THE STACK
010172  032737  000002  012174          BIT #BIT1,ASTAT         ;IS THIS A PARITY ERROR
010200  001413                  BEQ 1$                  ;NO, SEE IF ITS A CRC ERROR

                        ;PARITY ERROR DURING A READ FUNCTION
010202  012737  016572  007314          MOV #MREAD,PTYP1+2      ;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
010210  012737  010052  007352          MOV #REREAD,PLOOP+2     ;IF LOOP ON ERROR LOOP THROUGH PLOOP
010216  012737  010160  007356          MOV #NEXTRD,PCONT+2     ;IF NO LOOP GO TO NEXT READ
010224  000137  007300                  JMP PARTEST             ;RECORD PARITY ERROR AND RETRY FUNCTION
010230  032737  000001  012174  1$:     BIT #BIT0,ASTAT         ;IS THIS A CRC ERROR
010236  001011                  BNE CRCER               ;YES GO TEST AND LOG IT

                        ;SEEK ERROR DURING A READ FUNCTION
010240  012737  010052  007632          MOV #REREAD,SEKRTY+2    ;SET SEEK CONTINUE FOR READ RETRY
010246  012737  016572  007570          MOV #MREAD,STYP1+2      ;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
010254  004737  007554                  JSR PC,SEEKER           ;RECORD SEEK ERROR
010260  000737                  BR NEXTRD               ;GO TO NEXT SECTOR,CAN'T READ THIS ONE
```

```
                                        ;CRC ERROR DETECTED WHILE READING
                        CRCER:          TST R1                  ;IF READ ONLY, REPORT DATA CRC ERROR
                                        BPL DATACRC
                                        INC DATACK              ;SET DATA CHECK FLAG
                                        JSR PC,EMPBUFF          ;CHECK FOR A DATA ERROR
                                        TST ERCNTR              ;WAS THERE A DATA ERROR
                                        BNE DATACRC             ;YES, REPORT IT
                                        CKSWR
                                        BIT #SW13,@SWR          ;TEST DON'T PRINT SWITCH
                                        BNE 2$
                                        TYPE ,MBADCRC           ;TYPE CRC GENERATOR ERROR
                                        TYPE ,MCRLF
                        2$:             CKSWR
                                        TST @SWR                ;TEST HALT ON ERROR SWITCH
                                        BPL CONT15
                        HLT10:          HALT                    ;HALT ON ERROR
                        CONT15:         BIT #SW11,@SWR          ;CHECK LOOP ON ERROR SWITCH
                                        BNE 3$
                                        BR NEXTRD               ;DON'T LOOP GO TO NEXT SECTOR
                        3$:             JMP REREAD              ;LOOP ON TEST.

                                        ;DATA CRC ERROR

                        DATACRC:        CKSWR
                                        BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
                                        BNE 4$
                                        TYPE ,MCRC              ;TYPE DATA CRC ERROR
                                        TYPE ,MCRLF
                        4$:             CKSWR
                                        TST @SWR                ;TEST HALT ON ERROR SWITCH
                                        BPL CONT16
                        HLT12:          HALT                    ;HALT ON ERROR
                        CONT16:         BIT #SW11,@SWR          ;TEST LOOP ON ERROR
                                        BNE 5$                  ;IF SET LOOP ON THE TEST
                                        ADD #2,SP               ;REMOVE READ DONE ADDRESS FROM STACK
                                        JMP NEXTRD              ;READ NEXT SECTOR CAN'T READ THIS ONE
                        5$:             JMP REREAD              ;NO,GO REREAD THIS SECTOR
```

```
2971  010436  022737  006076  006604  DDCHK:      CMP #T25,PCSCOPE        ;IS THIS TEST 25
2972  010444  001041                              BNE CONT10
2973  010446  132737  000100  012174              BITB #BIT6,ASTAT        ;THIS IS TEST 25
2974  010454  001056                              BNE RETURN              ;DD BIT SHOULD BE SET
2975  010456  104406                              CKSWR
2976  010460  032777  020000  170530              BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
2977  010466  001013                              BNE CONT11
2978  010470  004737  010614                      JSR PC,ERMSG
2979  010474  104401  016011                      TYPE ,MDDMIS            ;TYPE MISSING DELETED DATA BIT
2980  010500  052737  000400  012752  DDERR:      BIS #BIT8,UNITSEL       ;SET HARD ERROR FLAG
2981  010506  004737  012076                      JSR PC,TYPADR           ;TYPE ADDRESS OF ERROR
2982  010512  104401  016162                      TYPE ,MCRLF
2983  010516  104406                  CONT11:     CKSWR
2984  010520  005777  170472                      TST @SWR                ;TEST HALT ON ERROR SWITCH
2985  010524  100001                              BPL CONT17
2986  010526  000000                  HLT13:      HALT                    ;HALT ON DELETED DATA ERROR
2987  010530  032777  004000  170460  CONT17:     BIT #SW11,@SWR          ;TEST LOOP ON ERROR
2988  010536  001402                              BEQ 4$
2989  010540  000137  010052                      JMP REREAD              ;LOOP ON TEST
2990  010544  000137  010160              4$:     JMP NEXTRD              ;READ NEXT SECTOR
2991  010550  032737  000100  012174  CONT10:     BIT #BIT6,ASTAT         ;THIS IS NOT A DELETED DATA TRANSFER
2992  010556  001415                              BEQ RETURN
2993  010560  052737  000400  012752              BIS #BIT8,UNITSEL       ;SET HARD ERROR FLAG
2994  010566  104406                              CKSWR
2995  010570  032777  020000  170420              BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
2996  010576  001347                              BNE CONT11
2997  010600  004737  010614                      JSR PC,ERMSG
2998  010604  104401  015763                      TYPE ,MUNXDD            ;TYPE UNEXPECTED DELETED DATA BIT
2999  010610  000733                              BR DDERR
3000  010612  000207                  RETURN:     RTS PC
3001
3002
3003  010614  104401  016165          ERMSG:      TYPE ,MERHEADER
3004  010620  013746  006604                  MOV     PCSCOPE,-(SP)   ;;SAVE PCSCOPE FOR TYPEOUT
3005  010624  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3006  010626     006                          .BYTE   6               ;;TYPE 6 DIGITS
3007  010627     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3008  010630  104401  006753                  TYPE ,MPASS
3009  010634  013737  002556  010646          MOV PASS,1$
3010  010642  004537  015642                  JSR R5,SGLDEC
3011  010646  000000                  1$:     OPEN
3012  010650  104401  016162                  TYPE ,MCRLF
3013  010654  004737  006476                  JSR PC,DING
3014  010660  000207                          RTS PC
3015
3016
```

```
                                        .SBTTL READ AND VERIFY DATA
                                        ;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
                                        ;THE DATA READ AGAINST CORE DATA BUFFER
010662  052701  100000        READCHK:  BIS #BIT15,R1          ;SET READ ONE SECTOR FLAG
010666  004737  010042                  JSR PC,READ            ;GO READ ONE SECTOR
010672  005737  013442        EMPBUFF:  TST SECCNTR            ;IF CLEARED NO SECTORS WERE FOUND
010676  001002                          BNE 1$
010700  000137  011474                  JMP EXIT               ;GO TO NEXT TRACK
010704  005037  011500        1$:       CLR BYTECNTR           ;CLEAR THE BYTE AND ERROR COUNTERS
010710  005037  011506                  CLR ERCNTR
010714  052701  000200                  BIS #BIT7,R1           ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
010720  004737  011014                  JSR PC,ADJSUM          ;ADJUST DATA AND CK SUM FOR ADDRESSES
010724  005037  011104                  CLR CKSUM              ;SET UP FOR CHECK SUM ACCUMULATION
010730  012746  011334                  MOV #EMPDONE,-(SP)     ;SET UP RETURN ADDRESSES
010734  012746  011106                  MOV #EMPER,-(SP)
010740  005046                  CLR     -(SP)                  ;LOWER 'CPU' LEVEL
010742  012746  010750          MOV     #2$,-(SP)              ;SET RETURN 'PC'
010746  000002                  RTI                            ;GET 'CPU' LEVEL INTO 'PSW'
010750  012777  000103  170230  2$:       MOV #EBIE,@RXCS        ;LOAD EMPTY BUFFER FUNCTION
010756  105777  170224        EMPFLAG:  TSTB @RXCS             ;TEST FOR TR FLAG
010762  100375                          BPL EMPFLAG

010764  117737  170220  011502  CKBYTE:   MOVB @RXDB,BADBYTE    ;SAVE BYTE FROM DISKETTE
010772  063737  011502  011104            ADD BADBYTE,CKSUM      ;ACCUMULATE CHECK SUM
011000  123720  011502                    CMPB BADBYTE,(R0)+     ;COMPARE AGAINST GOOD BYTE
011004  001054                            BNE DATAER             ;IF NOT EQUAL GO TO DATAER
011006  005237  011500                    INC BYTECNTR
011012  000761                          BR EMPFLAG             ;GET NEXT BYTE

011014  113737  013154  017456  ADJSUM:   MOVB TARGET,BUFADR    ;SET FIRST AND SECOND BYTES WITH ADDRESSES
011022  113737  013444  017457            MOVB TSECTOR,BUFADR+1
011030  013737  012564  011104            MOV SUM,CKSUM          ;GET THE PATTERN SUM
011036  063737  013154  011104            ADD TARGET,CKSUM       ;ADD TRACK ADDRESS TO CHECK SUM
011044  063737  013444  011104            ADD TSECTOR,CKSUM      ;ADD SECTOR ADDRESS TO CHECK SUM
011052  113737  011104  017654            MOVB CKSUM,BUFADR+176  ;INSERT CHECK SUM TO DATA BUFFER
011060  106337  011104                    ASLB CKSUM             ;GENERATE NEGITIVE CHECK SUM
011064  105437  011104                    NEGB CKSUM
011070  113737  011104  017655            MOVB CKSUM,BUFADR+177  ;INSERT NEG,SUM INTO DATA BUFFER
011076  012700  017456                    MOV #BUFADR,R0         ;SET ADDRESS OF BYTE IN R0
011102  000207                          RTS PC                 ;RETURN

011104  000000          CKSUM:    0

011106  005726          EMPER:    TST (SP)+                    ;REMOVE THE DONE RETURN FROM THE STACK
011110  012737  016444  007314            MOV #MEMPTY,P1YP1+2    ;PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYPOUT 1
011116  012737  010672  007352            MOV #EMPBUFF,PLOOP+2   ;RETURN THROUGH HERE TO LOOP ON ERROR
011124  012737  011456  007356            MOV #NXREAD,PCONT+2    ;IF NO LOOP ON ERROR GO TO NEXT SECTOR
011132  000137  007300                    JMP PARTEST            ;REPORT PARITY ERROR
```

```
3068  011136  052737  000400  012752  DATAER:     BIS #BIT8,UNITSEL       ;SET THE HAD ERROR FLAG
3069  011144  005237  011506                      INC ERCNTR              ;INC THE BYTE ERROR COUNTER
3070  011150  104406                              CKSWR
3071  011152  032777  020000  170036              BIT #SW13,@SWR          ;TEST PRINT ERROR SW IN SWR
3072  011160  001054                              BNE NOERTYP             ;DON'T PRINT THE ERROR
3073  011162  032777  001000  170026              BIT #SW9,@SWR           ;TEST PRINT 10 ERRORS SWITCH
3074  011170  001004                              BNE 1$                  ;IF SET PRINT ALL ERRORS
3075  011172  023727  011506  000012              CMP ERCNTR,#10.         ;HAVE 10 ERRORS BEEN TYPED
3076  011200  003044                              BGT NOERTYP             ;YES,DON'T PRINT ANY MORE
3077  011202  105701                      1$:     TSTB R1                 ;TEST FIRST ERROR FLAG
3078  011204  100014                              BPL TYPERR
3079  011206  004737  010614                      JSR PC,ERMSG            ;PRINT ADDRESS OF TEST
3080  011212  104401  016034                      TYPE ,MDERHDR           ;FIRST ERROR, PRINT ERROR HEADER
3081  011216  104401  016162                      TYPE ,MCRLF
3082  011222  004737  012076                      JSR PC,TYPADR           ;PRINT TRACK AND SECTOR LOCATIONS
3083  011226  104401  016123                      TYPE ,MCOLMUN           ;SET UP COLMUN HEADINGS
3084  011232  042701  000200                      BIC #BIT7,R1            ;CLEAR FIRST ERROR FLAG
3085  011236  013737  011500  011250  TYPERR:     MOV BYTECNTR,1$         ;PRINT BYTE NUMBER
3086  011244  004537  015642                      JSR R5,SGLDEC
3087  011250  000000                      1$:     OPEN
3088  011252  104401  016157                      TYPE ,DBLSP
3089  011256  013746  011502                      MOV BADBYTE,-(SP)       ;PRINT BYTE READ FROM DISKETTE
3090  011262  104403                              TYPOS
3091  011264  000003                              .WORD 3
3092  011266  104401  016157                      TYPE ,DBLSP
3093  011272  114037  011504                      MOVB -(R0),GOODBYTE     ;GET GOOD BYTE
3094  011276  005200                              INC R0                  ;RETURN R0 TO NEXT BYTE IN BUFFER
3095  011300  013746  011504                      MOV GOODBYTE,-(SP)
3096  011304  104404                              TYPON                   ;PRINT GOOD DATA
3097  011306  104401  016162                      TYPE ,MCRLF
3098  011312  104406              NOERTYP:        CKSWR
3099  011314  005777  167676                      TST @SWR                ;TEST HALT ON ERROR SWITCH
3100  011320  100001                              BPL CONT20
3101  011322  000000              HLT14:          HALT
3102  011324  005237  011500      CONT20:         INC BYTECNTR
3103  011330  000137  010756                      JMP EMPFLAG
3104
```

```
3105  011334  005737  007732          EMPDONE:    TST DATACK          ;WAS THIS READ CHECK CAUSED BY A CRC ERROR
3106  011340  001401                              BEQ 1$              ;NO
3107  011342  000207                              RTS PC              ;YES,RETURN TO CRC HANDLER
3108  011344  005737  011506          1$:         TST ERCNTR          ;WAS THERE ERRORS
3109  011350  001442                              BEQ NXREAD          ;NO ERRORS
3110  011352  104406                              CKSWR
3111  011354  032777  020000  167634              BIT #SW13,@SWR      ;YES,TEST DON'T PRINT SWITCH
3112  011362  001024                              BNE 2$              ;DON'T PRINT THE ERROR
3113  011364  104401  016375                      TYPE ,MERCT         ;PRINT THE TOTAL DATA ERROR COUNT
3114  011370  013737  011506  011402              MOV ERCNTR,3$
3115  011376  004537  015642                      JSR R5,SGLDEC
3116  011402  000000          3$:                 OPEN
3117  011404  104401  016712                      TYPE ,MSUM          ;INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
3118  011410  105737  011104                      TSTB CKSUM
3119  011414  001403                              BEQ 4$
3120  011416  104401  016677                      TYPE ,MBAD
3121  011422  000402                              BR 5$
3122  011424  104401  016705          4$:         TYPE,MGOOD
3123  011430  104401  016162          5$:         TYPE ,MCRLF
3124  011434  104406          2$:                 CKSWR
3125  011436  032777  004000  167552              BIT #SW11,@SWR      ;TEST LOOP ON ERROR SWITCH
3126  011444  001404                              BEQ NXREAD          ;IF NOT SET GO TO NEXT SECTOR
3127  011446  004737  010052                      JSR PC,REREAD       ;YES,GO REREAD THE DATA
3128  011452  000137  010672                      JMP EMPBUFF         ;GO RECHECK THE DATA
3129  011456  005337  013442          NXREAD:     DEC SECCNTR
3130  011462  001404                              BEQ EXIT
3131  011464  004737  010046                      JSR PC,XREAD        ;READ THE NEXT SECTOR
3132  011470  000137  010672                      JMP EMPBUFF
3133  011474  005001          EXIT:               CLR R1              ;CLEAR THE ONE READ FLAG
3134  011476  000207                              RTS PC
3135
3136  011500  000000          BYTECNTR:           0
3137  011502  000000          BADBYTE:            0
3138  011504  000000          GOODBYTE:           0
3139  011506  000000          ERCNTR:             0
3140
3141                          ;;******************************************************************
3142
3143                                  ;AN INTERRUPT DID NOT OCCURE AT A FUNCTION DONE FLAG.
3144
3145  011510  104406          NOINTER:            CKSWR
3146  011512  032777  020000  167476              BIT #SW13,@SWR      ;TEST DON'T PRINT ERROR SWITCH
3147  011520  001006                              BNE 1$
3148  011522  004737  010614                      JSR PC,ERMSG
3149  011526  104401  016320                      TYPE ,MINTER        ;TYPE NO INTERRUPT ON DONE ERROR
3150  011532  104401  016162                      TYPE ,MCRLF
3151  011536  104406          1$:                 CKSWR
3152  011540  005777  167452                      TST @SWR            ;TEST HALT ON ERROR SWITCH
3153  011544  100001                              BPL CONT21
3154  011546  000000          HLT15:              HALT                ;HALT ON ERROR
3155  011550  004737  011554          CONT21:     JSR PC,INTSERV      ;JSR TO INTSERV AS IF IT WAS AN INTERRUPT
3156
```

```
                                        .SBTTL  INTERRUPT SERVICE

011554  117737  167430  012174  INTSERV:        MOVB @RXDB,ASTAT        ;SAVE THE ERROR AND STATUS WORD
011562  005777  167420                          TST @RXCS               ;TEST THE ERROR FLAG
011566  100444                                  BMI RXERROR             ;THERE WAS AN ERROR GO REPORT IT
011570  032737  000004  012174                  BIT #BIT2,ASTAT         ;IS INIT DONE SET
011576  001402                                  BEQ 2$                  ;NO,CONTINUE
011600  000137  012034                          JMP RXPWR               ;YES,REPORT POWER FAILED AND RESTART
011604  032737  000003  012174  2$:             BIT #3,ASTAT            ;ARE PAR OR CRC BITS SET
011612  001021                                  BNE 1$                  ;YES GO REPORT ERROR
011614  132777  000040  167364                  BITB #DONEBIT,@RXCS     ;IS DONE SET
011622  001012                                  BNE 3$                  ;IF SET RETURN TO TEST
011624  104406                                  CKSWR
011626  032777  020000  167362                  BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
011634  001004                                  BNE 4$                  ;DON'T PRINT
011636  104401  016353                          TYPE ,MUKNINT           ;TYPE UNKNOWN INTERRUPT
011642  104401  016162                          TYPE ,MCRLF
011646  000002                  4$:             RTI                     ;RETURN FROM THE INTERRUPT
011650  062706  000006          3$:             ADD #6,SP               ;BYPASS INTERRUPT POINTERS ON STACK
011654  000207                                  RTS PC                  ;RETURN TO PROGRAM
011656  104406                  1$:             CKSWR
011660  032777  020000  167330                  BIT #SW13,@SWR          ;TEST DON'T PRINT ERROR SWITCH
011666  001004                                  BNE RXERROR
011670  104401  016656                          TYPE ,MNOFLAG           ;TYPE NO STATUS ERROR ERROR
011674  104401  016162                          TYPE ,MCRLF
011700  005237  006520          RXERROR:        INC ERRORS              ;AN ERROR INDICATOR
011704  001775                                  BEQ RXERROR
011706  052737  000400  012752                  BIS #BIT8,UNITSEL       ;SET HARD ERROR FLAG
011714  012777  000017  167264  2$:             MOV #RDER,@RXCS         ;GET THE ERROR CODE
011722  004737  006622          3$:             JSR PC,SDN              ;TEST FOR DONE FLAG
011726  000775                                  BR 3$
011730  032777  000002  167252                  BIT #2,@RXDB            ;WAS THERE A PARITY ERROR
011736  001403                                  BEQ 1$                  ;NO,CONTINUE
011740  004737  012050                          JSR PC,PARTYP           ;YES,GO REPORT THE PARITY ERROR
011744  000763                                  BR 2$                   ;REISSUE THE FUNCTION
011746  117737  167236  012176  1$:             MOVB @RXDB,BSTAT        ;SAVE THE ERROR CODE IN B STATUS
011754  104406          NOPRNT:                 CKSWR
011756  032777  020000  167232                  BIT #SW13,@SWR          ;TEST PRINT ERROR SWITCH IN SWR
011764  001020                                  BNE 2$
011766  104401  016162                          TYPE ,MCRLF
011772  004737  010614                          JSR PC,ERMSG            ;TYPE ERROR AND MESSAGES
011776  104401  016254                          TYPE ,MRXCS             ;TYPE COMMAND STATUS REGISTER
012002  013746  007730                  MOV     FUNCTION,-(SP)  ;;SAVE FUNCTION FOR TYPEOUT
012006  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
012010     006                          .BYTE   6               ;;TYPE 6 DIGIT(S)
012011     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
012012  004737  012076                          JSR PC,TYPADR           ;TYPE ADDRESSES AND RUN CONDITIONS
012016  104401  016162                          TYPE ,MCRLF
012022  004737  012246                          JSR PC,TYPCODE          ;PRINT THE STATUS REGISTERS
012026  062706  000004          2$:             ADD #4,SP               ;MOVE ERROR RETURN TO TOP OF STACK
012032  000207                                  RTS PC

012034  104401  016727          RXPWR:          TYPE ,MRX11             ;ONLY THE RX11 POWER HAS FAILED
012040  104401  015332                          TYPE ,$POWER            ;PRINT POWER FAILED
012044  000137  001350                          JMP RESTART             ;GO TO RESTART
```

```
3212  012050  104401  016254          PARTYP:         TYPE ,MRXCS
3213  012054  017746  167126                  MOV     @RXCS,-(SP)     ;;SAVE @RXCS FOR TYPEOUT
3214  012060  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3215  012062     006                          .BYTE   6               ;;TYPE 6 DIGIT(S)
3216  012063     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3217  012064  104401  016641                  TYPE ,MPAR
3218  012070  104401  016162                  TYPE ,MCRLF
3219  012074  000207                          RTS PC
3220
3221  012076  104401  016062          TYPADR:         TYPE ,MTRK              ;TYPE TRACK ADDRESS
3222  012102  013746  013154                  MOV     TARGET,-(SP)    ;;SAVE TARGET FOR TYPEOUT
3223  012106  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3224  012110     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
3225  012111     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3226  012112  104401  016111                  TYPE ,MSECT             ;TYPE SECTOR ADDRESS
3227  012116  013737  013444  012172          MOV TSECTOR,2$
3228  012124  042737  177740  012172          BIC #177740,2$          ;CLEAR ALL BUT SECTOR ADDRESS
3229  012132  013746  012172                  MOV     2$,-(SP)        ;;SAVE 2$ FOR TYPEOUT
3230  012136  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3231  012140     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3232  012141     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3233  012142  104401  016157                  TYPE ,DBLSP
3234  012146  032737  000020  012752          BIT #BIT4,UNITSEL       ;WHITCH DRIVE IS BEING USED
3235  012154  001003                          BNE 1$
3236  012156  104401  016225                  TYPE ,MUNIT0            ;TYPE UNIT 0
3237  012162  000402                          BR 4$
3238  012164  104401  016235          1$:             TYPE ,MUNIT1            ;TYPE UNIT 1
3239  012170  000207                  4$:             RTS PC
3240  012172  000000                  2$:             OPEN
3241
3242  012174  000000                  ASTAT:          0
3243  012176  000000                  BSTAT:          0
3244
3245
3246  012200  117737  167004  012174  RDCODE:         MOVB @RXDB,ASTAT        ;SAVE THE A STATUS
3247  012206  012777  000017  166772  2$:             MOV #RDER,@RXCS         ;READ THE B STATUS REGISTER
3248  012214  004737  006622          3$:             JSR PC,SDN              ;WAIT FOR DONE FLAG
3249  012220  000775                          BR 3$
3250  012222  032777  000002  166760          BIT #2,@RXDB            ;WAS THERE A PARITY ERROR
3251  012230  001403                          BEQ 1$                  ;NO,CONTINUE
3252  012232  004737  012050                  JSR PC,PARTYP           ;YES,REPORT THE PARITY ERROR
3253  012236  000763                          BR 2$                   ;RETRY READING STATUS B
3254  012240  117737  166744  012176  1$:             MOVB @RXDB,BSTAT        ;SAVE THE B STATUS CODES
3255  012246  104401  016264          TYPCODE:        TYPE ,MASTAT            ;TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
3256  012252  013746  012174                  MOV     ASTAT,-(SP)     ;;SAVE ASTAT FOR TYPEOUT
3257  012256  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3258  012260     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
3259  012261     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3260  012262  104401  016150                  TYPE ,TAB
3261  012266  104401  016300                  TYPE ,MBSTAT
3262  012272  013746  012176                  MOV BSTAT,-(SP)
3263  012276  104404                          TYPON
3264  012300  104401  016162                  TYPE ,MCRLF
3265  012304  000207                          RTS PC
```

```
                        .SBTTL  PATTERN GENERATOR

                        ;NOTE: ALL DATA PATTERNS WILL BE MODIFIED SO THE FIRST BYTE WILL
                        ;CONTAIN THE TRACK ADDRESS. THE SECOND BYTE WILL CONTAIN THE UNIT
                        ;NUMBER AND SECTOR ADDRESS IN WHICH THE DATA IS WRITTEN. THE MOST
                        ;SIGNIFICANT BIT OF THIS SECOND BYTE INDICATES THE UNIT. UNIT 0
                        ;IF "0" UNIT 1 IF "1". THE LAST TWO BYTES CONTAIN THE CHECK SUM.
                    ;;**********************************************************************


012306  012704  017456      GETPATTERN:     MOV #BUFADR,R4          ;SET ADDRESS OF FIRST DATA BYTE
012312  005037  012564                      CLR SUM                 ;SET UP FOR ACCUMULATION OF CHECK SUM
012316  013705  012352                      MOV PAT,R5              ;GET PATTERN BITS
012322  006305                              ASL R5
012324  000175  012330                      JMP @PATTERNS(R5)
012330  012354              PATTERNS:       DATA0                   ;000 DATA BYTE
012332  012366                              DATA1                   ;377 DATA BYTE
012334  012376                              FLOAT0                  ;FLOAT A 0 THROUGH ALL 1'S
012336  012440                              FLOAT1                  ;FLOAT A 1 THROUGH ALL 0'S
012340  012446                              PAT125                  ;125/052 DATA WORD
012342  012466                              PAT314                  ;314/063 DATA WORD
012344  012476                              COUNT                   ;INCRUMENT DATA PATTERN
012346  012516                              RANDATA                 ;RANDOM DATA BYTE

012350  000000              DATABYTE:       0
012352  000000              PAT:            0

                    ;;**********************************************************************

                            ;LOAD SOFTWARE BUFFER WITH ALL ZEROS
                            ;   PAT = 0

012354  005037  012350      DATA0:          CLR DATABYTE
012360  004737  012536      PATGEN:         JSR PC,LOAD             ;GO LOAD THE DATA BUFFER
012364  000775                              BR PATGEN

                    ;;**********************************************************************

                            ;LOAD SOFTWARE BUFFER WITH ALL ONES
                            ;   PAT = 1

012366  112737  000377  012350  DATA1:      MOVB #377,DATABYTE
012374  000771                              BR PATGEN
```

```
                                        ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
                                        ;   PAT = 2
                                        ;
012376  112737  000376  012350  FLOAT0:       MOVB #376,DATABYTE      ;SET UP A ONES FIELD
012404  000261                  XPATGEN:      SEC                     ;SET THE C BIT TO ROTATE THROUGH THE DATA
012406  012702  000000          1$:           MOV #0,R2               ;CLR R2 (CAN'T USE "CLR" IT CLEARS "C" BIT)
012412  103001                                BCC 2$                  ;BR IF "C" BIT IS CLEARED
012414  005202                                INC R2                  ;SET R2 IF "C" BIT IS SET
012416  004737  012536          2$:           JSR PC,LOAD             ;GO LOAD THE DATA BUFFER
012422  000241                                CLC                     ;CLEAR THE "C" BIT
012424  005702                                TST R2                  ;IS R2 NONZERO
012426  001401                                BEQ 3$
012430  000261                                SEC                     ;YES, SET THE "C" BIT
012432  106137  012350          3$:           ROLB DATABYTE
012436  000763                                BR 1$
                                        ;
                                        ;;*************************************************************************
                                        ;
                                        ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
                                        ;   PAT = 3
                                        ;
012440  005037  012350          FLOAT1:       CLR DATABYTE
012444  000757                                BR XPATGEN
                                        ;
                                        ;;*************************************************************************
                                        ;
                                        ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
                                        ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
                                        ;   PAT = 4
                                        ;
012446  112737  000125  012350  PAT125:       MOVB #125,DATABYTE
012454  004737  012536          XXPATGEN:     JSR PC,LOAD
012460  105137  012350                        COMB DATABYTE
012464  000773                                BR XXPATGEN
                                        ;
                                        ;;*************************************************************************
                                        ;
                                        ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
                                        ;COMPLIMENT INTO THE NEXT
                                        ;   PAT = 5
                                        ;
012466  112737  000314  012350  PAT314:       MOVB #314,DATABYTE
012474  000767                                BR XXPATGEN
                                        ;
                                        ;;*************************************************************************
                                        ;
                                        ;LOAD SOFTWARE BUFFER WITH COUNT PATTERN
                                        ;   PAT = 6
                                        ;
012476  012737  000377  012350  COUNT:        MOV #377,DATABYTE
012504  005237  012350          1$:           INC DATABYTE
012510  004737  012536                        JSR PC,LOAD
012514  000773                                BR 1$
```

```
3367                                    ;;*****************************************************************
3368
3369                                            ;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
3370                                            ;   PAT = 7
3371
3372  012516  004737  012566              RANDATA:        JSR PC,RANGEN           ;GET RANDOM NUMBER
3373  012522  113737  012660  012350                      MOVB RANUM,DATABYTE
3374  012530  004737  012536                              JSR PC,LOAD
3375  012534  000770                                      BR RANDATA
3376
3377  012536  063737  012350  012564  LOAD:           ADD DATABYTE,SUM        ;ACCUMULATE THE PATTERN CHECK SUM
3378  012544  113724  012350                          MOVB DATABYTE,(R4)+     ;LOAD THE DATA BUFFER
3379  012550  022704  017656                          CMP #BUFADR+200,R4      ;HAVE 128 BYTES BEEN GENERATED
3380  012554  001401                                  BEQ 1$                  ;IF YES,RETURN TO TEST
3381  012556  000207                                  RTS PC                  ;IF NO,RETURN TO PATTERN GENERATOR
3382  012560  005726                          1$:     TST (SP)+               ;TAKE PATTERN RETURN ADDRESS OF STACK
3383  012562  000207                                  RTS PC                  ;RETURN TO TEST
3384
3385  012564  000000                          SUM:    0
3386
3387  012566  012700  000001              RANGEN:         MOV #1,R0
3388  012572  063700  012654                              ADD RAN1,R0
3389  012576  063700  012656                              ADD RAN2,R0
3390  012602  042700  170000                              BIC #170000,R0
3391  012606  000241                                      CLC
3392  012610  006100                                      ROL R0
3393  012612  006100                                      ROL R0
3394  012614  010037  012654                              MOV R0,RAN1
3395  012620  005000                                      CLR R0
3396  012622  013700  012656                              MOV RAN2,R0
3397  012626  006000                                      ROR R0
3398  012630  006000                                      ROR R0
3399  012632  063700  012654                              ADD RAN1,R0
3400  012636  042700  170000                              BIC #170000,R0
3401  012642  010037  012656                              MOV R0,RAN2
3402  012646  010037  012660                              MOV R0,RANUM
3403  012652  000207                                      RTS PC
3404
3405  012654  001234                  RAN1:           001234
3406  012656  000765                  RAN2:           000765
3407  012660  000000                  RANUM:          0
3408
```

```
                                        .SBTTL  UNIT SELECTION

                                        ;TEST FOR SELECTED UNITS,DRIVE READY,AND USED CONDITIONS
                                        ;ALSO CONTAINS A "HAD ERROR" FLAG TO BE TESTED AT EOP.
                                        ;THE BITS IN UNITSEL ARE USED AS FOLLOWS
                                        ;
                                        ;BIT15  =UNIT 1 SELECTED VIA SWR
                                        ;BIT14  =UNIT 1 USED BIT
                                        ;BIT8   =THIS PASS HAD AN ERROR
                                        ;BIT7   =UNIT 0 SELECTED VIA SWR
                                        ;BIT6   =UNIT 0 USED BIT
                                        ;BIT4   =UNIT SELECTION FOR FUNCTION WORD

                                    ;;**********************************************************************
012662  032737  000100  012752  GETUNIT:    BIT #BIT6,UNITSEL       ;WAS UNIT 0 JUST USED
012670  001012                              BNE 1$                  ;UNIT 0 USED CHECK UNIT 1
012672  105737  012752                      TSTB UNITSEL            ;WAS UNIT 0 SELECTED
012676  100007                              BPL 1$                  ;NO GO TO UNIT 1
012700  042737  040020  012752              BIC #40020,UNITSEL      ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
012706  052737  000100  012752              BIS #BIT6,UNITSEL       ;SET UNIT 0 USED BIT
012714  000207                              RTS PC
012716  005737  012752          1$:         TST UNITSEL             ;WAS UNIT 1 SELECTED
012722  100012                              BPL 2$                  ;NO RETURN
012724  032737  040000  012752              BIT #BIT14,UNITSEL      ;HAS UNIT 1 BEEN USED
012732  001006                              BNE 2$                  ;YES RETURN
012734  042737  000100  012752              BIC #BIT6,UNITSEL       ;CLEAR UNIT 0 USED BIT
012742  052737  040020  012752              BIS #40020,UNITSEL      ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
012750  000207                      2$:     RTS PC


012752  000000              UNITSEL:        0

                                    ;TEST THAT ALL UNITS HAVE BEEN ACCESSED

012754  005737  012752      DONE:           TST UNITSEL             ;IS UNIT 1 SELECTED
012760  100006                              BPL 1$                  ;NO RETURN
012762  032737  040000  012752              BIT #BIT14,UNITSEL      ;YES HAS IT BEEN USED
012770  001002                              BNE 1$                  ;YES RETURN
012772  062716  000002                      ADD #2,@SP              ;BYPASS NOT DONE RETURE ON STACK
012776  000207                      1$:     RTS PC
```

```
                                        .SBTTL  TRACK SEQUENCE SELECTION

                                        ;INITIALIZE TRACK SEQUENCE

                                ;NOTE: IF WORD SEQUEN IS CLEARED THEN TRACK SEQUENCE IS FROM 0-52-53-114 ONLY
                                ;IF BIT 15 OF SEQUEN IS "1" THEN TRACK SELECTION IS INC. BETWEEN SELECTED OD/ID LIMITS.
                                ;IF BIT 7 IS "1" THEN TEST 25 DECREMENT SEQUENCE IS REQUIRED.


013000  105737  013164        INITTRACK:      TSTB SEQUEN             ;IS THIS TEST 26 SPECIAL SEQUENCE
013004  100442                                 BMI 2$                  ;YES, DEC FROM TRACK 12 TO 0
013006  042737  100200  001200                BIC #100200,OD          ;CLEAR FIRST USED BITS
013014  005737  001200                        TST OD                  ;TEST CONTENTS OF ID,OD FOR 0
013020  001440                                 BEQ 3$                  ;SEQUENCE WILL BE FROM "HOME"-52-53-114-0
013022  052737  100000  013164                BIS #BIT15,SEQUEN       ;LIMITS WERE SELECTED, INC FROM OD TO ID.
013030  113737  001200  013154                MOVB OD,TARGET          ;INIT OD AS PRESENT TRACK
013036  005037  013162                        CLR XID                 ;INIT WORKING ID AND OD LOCATIONS
013042  113737  001201  013162                MOVB ID,XID
013050  005037  013160                        CLR XOD
013054  113737  001200  013160                MOVB OD,XOD
013062  013737  013162  013152                MOV XID,TRKCNTR         ;SET UP NUMBER OF TRACK MOVEMENTS
013070  163737  013160  013152                SUB XOD,TRKCNTR
013076  005237  013152                        INC TRKCNTR
013102  052737  100200  001200  1$:           BIS #100200,OD          ;SET FIRST TIME BITS IN ID,OD
013110  000207                                 RTS PC
013112  012737  000013  013152  2$:           MOV #13,TRKCNTR         ;SET TRACK COUNTER
013120  000770                                 BR 1$
013122  012737  000004  013152  3$:           MOV #4,TRKCNTR          ;SET THE TRACK COUNTER
013130  000764                                 BR 1$


                                ;;**************************************************************

013132  113737  013154  013156  GETTRACK:      MOVB TARGET,PRESTRK     ;RESET TO PRESENT TRACK
013140  005737  013164                        TST SEQUEN              ;IS THIS THE LIMITED SEQUENCE
013144  001410                                 BEQ LIMTRK              ;YES, DOING ONLY 0-52-53-114
013146  100446                                 BMI SEQ1                ;NO,SEQUENCE IS BETWEEN SELECTED LIMITS
013150  000463                                 BR SEQ2                 ;NO,THIS IS TEST 26 DEC SEQUENCE

013152  000000                TRKCNTR:        0
013154  000000                TARGET:         0
013156  000000                PRESTRK:        0
013160  000000                XOD:            0
013162  000000                XID:            0
013164  000000                SEQUEN:         0
```

```
                                ;;*****************************************************************

                                        ;LIMITED SEQUENCE, ACCESS TRACKS 52 TO 53 TO 114 BACK TO 0
013166  005737  001200          LIMTRK:       TST OD                  ;TEST HIGH ORDER FIRST TIME BIT
013172  100007                              BPL 1$                  ;NOT SET, ON TRACK 52
013174  012737  000052  013154              MOV #52,TARGET          ;GO TO TRACK 52
013202  042737  100000  001200              BIC #BIT15,OD           ;CLEAR FIRST TIME BIT
013210  000207                              RTS PC
013212  105737  001200          1$:         TSTB OD                 ;TEST LOW ORDER FIRST TIME BIT
013216  100007                              BPL 2$                  ;NOT SET, ON TRACK 53
013220  012737  000053  013154              MOV #53,TARGET          ;GO TO TRACK 53
013226  042737  000200  001200              BIC #BIT7,OD
013234  000207                              RTS PC
013236  023727  013154  000114  2$:         CMP TARGET,#114         ;IS IT ON TRACK 114
013244  001404                              BEQ 3$                  ;YES,GO TO TRACK 0
013246  012737  000114  013154              MOV #114,TARGET         ;NO, GO TO TRACK 114
013254  000207                              RTS PC
013256  005037  013154          3$:         CLR TARGET              ;GO TO TRACK 0
013262  000207                              RTS PC

                                ;;*****************************************************************

                                        ;INCREMENT FROM OD+1 TO ID AND RETURN TO OD
                                        ; USED WHEN TRACK LIMITS ARE SELECTED

013264  042737  100200  001200  SEQ1:       BIC #100200,OD          ;CLEAR FIRST TIME BITS
013272  123737  013162  013156              CMPB XID,PRESTRK        ;PRESENT TRACK EQUAL TO ID
013300  001004                              BNE 1$                  ;NO GET NEW TRACK
013302  113737  001200  013154              MOVB OD,TARGET          ;YES RETURN TO OD
013310  000207                              RTS PC
013312  005237  013154          1$:         INC TARGET              ;ADD 1 TO TARGET TRACK
013316  000207                              RTS PC

                                ;;*****************************************************************

                                        ;DECREMENT FROM ID = 12 TO OD = 0
                                        ;USED IN TEST 26 ONLY

013320  005737  001200  SEQ2:       TST OD                  ;FIRST TIME BIT SET
013324  100007                              BPL 1$                  ;NO GET NEXT TRACK
013326  042737  100200  001200              BIC #100200,OD          ;YES CLEAR FIRST TIME BITS
013334  012737  000012  013154              MOV #12,TARGET          ;MOVE OUT 10 TRACKS
013342  000207                              RTS PC
013344  005337  013154          1$:         DEC TARGET              ;MOVE TO NEXT TRACK
013350  000207                              RTS PC
```

```
                                        .SBTTL  SECTOR SELECTION

                                        ;SECTOR INITIALIZATION AND SELECTION

013352  005737  001202        INITSECTOR:     TST FIRST               ;TEST FIRST AND LAST FOR 0
013356  001005                                BNE 1$                  ;SECTORS SPECIFIED USE THEM
013360  005237  001202                        INC FIRST               ;NONE SPECIFIED SET FIRST TO 1
013364  112737  000032  001203                MOVB #32,LAST           ;SET LAST TO MAXIMUM
013372  113737  001203  013442  1$:           MOVB LAST,SECCNTR       ;SET UP SECTOR COUNTER
013400  163737  001202  013442                SUB FIRST,SECCNTR
013406  005237  013442                        INC SECCNTR
013412  105037  013443                        CLRB SECCNTR+1
013416  113737  001202  013444                MOVB FIRST,TSECTOR      ;PUT FIRST SECTOR IN TARGET SECTOR
013424  162737  000003  013444                SUB #3,TSECTOR          ;SUB 3 FROM TSECTOR AS FIRST TIME THROUGH
                                                                      ;IT GETS ADDED BACK ON.
013432  012737  000001  013446                MOV #1,INTLEAV          ;SET INTERLEAVE OFFSET
013440  000207                                RTS PC

013442  000000                SECCNTR:        0
013444  000000                TSECTOR:        0
013446  000000                INTLEAV:        0

013450  042737  000200  013444  GETSECTOR:    BIC #200,TSECTOR        ;CLEAR THE UNIT BIT BEFORE TESTING
013456  062737  000003  013444                ADD #3,TSECTOR          ;ADD 3 FOR INTERLEAVING
013464  123737  001203  013444                CMPB LAST,TSECTOR
013472  002010                                BGE 1$                  ;NEW SECTOR IS WITHIN LIMITS
013474  113737  001202  013444                MOVB FIRST,TSECTOR      ;RESET TARGET SECTOR TO INTERLEAVE
013502  063737  013446  013444                ADD INTLEAV,TSECTOR     ;ADD ON INTERLEAVE OFFSET VALUE
013510  005237  013446                        INC INTLEAV             ;UP DATE THE OFFSET VALUE
013514  032737  000020  012752  1$:           BIT #BIT4,UNITSEL       ;IS THIS UNIT 0
013522  001403                                BEQ 2$
013524  052737  000200  013444                BIS #BIT7,TSECTOR       ;NO, SET UNIT IDENTIFIER IN TARGET SECTOR
013532  000207                2$:             RTS PC
```

```
                        .SBTTL  TYPE ROUTINE

                        ;;*****************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*

013534  105737  013763  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
013540  100002                  BPL     1$              ;;BR IF YES
013542  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
013544  000407                  BR      3$              ;;LEAVE
013546  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
013550  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
013554  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
013556  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
013560  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
013562  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
013564  062716  000002  3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
013570  000002                  RTI                     ;;RETURN
013572  122716  000011  4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
013576  001430                  BEQ     8$
013600  122716  000200          CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
013604  001006                  BNE     5$
013606  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
013610  104401                  TYPE                    ;;TYPE A CR AND LF
013612  013765                  $CRLF
013614  105037  013750          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
013620  000755                  BR      2$              ;;GET NEXT CHARACTER
013622  004737  013704  5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
013626  123726  013762  6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
013632  001350                  BNE     2$              ;;IF NO GO GET NEXT CHAR.
013634  013746  013760          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                        ;;AND THE NULL CHAR.
013640  105366  000001  7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
013644  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
013646  004737  013704          JSR     PC,$TYPEC       ;;GO TYPE A NULL
013652  105337  013750          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
013656  000770                  BR      7$              ;;LOOP

                        ;HORIZONTAL TAB PROCESSOR

013660  112716  000040  8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
013664  004737  013704  9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
013670  132737  000007  013750  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
013676  001372                  BNE     9$              ;;TAB STOP
013700  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
```

```
3633  013702  000724                        BR      2$              ;;GET NEXT CHARACTER
3634  013704  105777  000044        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3635  013710  100375                        BPL     $TYPEC
3636  013712  116677  000002  000036      MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3637  013720  122766  000015  000002      CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3638  013726  001003                        BNE     1$              ;;BRANCH IF NO
3639  013730  105037  013750              CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3640  013734  000406                        BR      $TYPEX          ;;EXIT
3641  013736  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3642  013744  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
3643  013746  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
3644  013750  000000        $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3645  013752  000207        $TYPEX: RTS     PC
3646
3647  013754  177564        $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
3648  013756  177566        $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
3649  013760     000        $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
3650  013761     002        $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
3651  013762     012        $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
3652  013763     000        $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
3653  013764     077        $QUES:  .ASCII  "?"             ;;QUESTION MARK
3654  013765     015        $CRLF:  .ASCII  <15>            ;;CARRAIGE RETURN
3655  013766  000012        $LF:    .ASCIZ  <12>            ;;LINEFEED
```

```
                                                    1  9
3656                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3657
3658                                    ;;*****************************************************************
3659                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3660                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3661                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3662                                    ;*CALL:
3663                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3664                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
3665                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3666                                    ;*      .BYTE   M               ;;M=1 OR 0
3667                                    ;*                                      ;;1=TYPE LEADING ZEROS
3668                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
3669                                    ;*
3670                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3671                                    ;*$TYPOS OR $TYPOC
3672                                    ;*CALL:
3673                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3674                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
3675                                    ;*
3676                                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3677                                    ;*CALL:
3678                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3679                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
3680
3681  013770  017646  000000           $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
3682  013774  116637  000001  014213           MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
3683  014002  112637  014215                   MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3684  014006  062716  000002                   ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
3685  014012  000406                           BR      $TYPON
3686  014014  112737  000001  014213 $TYPOC:  MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
3687  014022  112737  000006  014215           MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
3688  014030  112737  000005  014212 $TYPON:  MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
3689  014036  010346                           MOV     R3,-(SP)        ;;SAVE R3
3690  014040  010446                           MOV     R4,-(SP)        ;;SAVE R4
3691  014042  010546                           MOV     R5,-(SP)        ;;SAVE R5
3692  014044  113704  014215                   MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
3693  014050  005404                           NEG     R4
3694  014052  062704  000006                   ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3695  014056  110437  014214                   MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3696  014062  113704  014213                   MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
3697  014066  016605  000012                   MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3698  014072  005003                           CLR     R3              ;;CLEAR THE OUTPUT WORD
3699  014074  006105                   1$:     ROL     R5              ;;ROTATE MSB INTO "C"
3700  014076  000404                           BR      3$              ;;GO DO MSB
3701  014100  006105                   2$:     ROL     R5              ;;FORM THIS DIGIT
3702  014102  006105                           ROL     R5
3703  014104  006105                           ROL     R5
3704  014106  010503                           MOV     R5,R3
3705  014110  006103                   3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3706  014112  105337  014214                   DECB    $OMODE          ;;TYPE THIS DIGIT?
3707  014116  100016                           BPL     7$              ;;BR IF NO
3708  014120  042703  177770                   BIC     #177770,R3      ;;GET RID OF JUNK
3709  014124  001002                           BNE     4$              ;;TEST FOR 0
3710  014126  005704                           TST     R4              ;;SUPPRESS THIS 0?
3711  014130  001403                           BEQ     5$              ;;BR IF YES
```

```
3712  014132  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3713  014134  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3714  014140  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3715  014144  110337  014210                  MOVB    R3,8$           ;;SAVE FOR TYPING
3716  014150  104401  014210                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
3717  014154  105337  014212          7$:     DECB    $OCNT           ;;COUNT BY 1
3718  014160  003347                          BGT     2$              ;;BR IF MORE TO DO
3719  014162  002402                          BLT     6$              ;;BR IF DONE
3720  014164  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3721  014166  000744                          BR      2$              ;;GO DO THE LAST DIGIT
3722  014170  012605          6$:             MOV     (SP)+,R5        ;;RESTORE R5
3723  014172  012604                          MOV     (SP)+,R4        ;;RESTORE R4
3724  014174  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3725  014176  016666  000002  000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3726  014204  012616                          MOV     (SP)+,(SP)
3727  014206  000002                          RTI                     ;;RETURN
3728  014210     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3729  014211     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3730  014212     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3731  014213     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3732  014214  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

                                ;;******************************************************************
                                ;*SAVE R0-R5
                                ;*CALL:
                                ;*      SAVREG
                                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                ;*
                                ;*TOP---(+16)
                                ;* +2---(+18)
                                ;* +4---R5
                                ;* +6---R4
                                ;* +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0

014216                          $SAVREG:
014216  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
014220  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014222  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
014224  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014226  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014230  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
014232  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
014236  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
014242  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
014246  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
014252  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
014254                          $RESREG:
014254  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
014260  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
014264  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
014270  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
014274  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
014276  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
014300  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
014302  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
014304  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
014306  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
014310  000002                          RTI
```

```
                        .SBTTL  TTY INPUT ROUTINE

                        ;;*****************************************************************
014312  177560         $TKS:   .WORD   177560          ;;TTY KBD STATUS
014314  177562         $TKB:   .WORD   177562          ;;TTY KBD BUFFER
                        .ENABL  LSB

                        ;;*****************************************************************
                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                        ;*WHEN OPERATING IN TTY FLAG MODE.
014316  022737  000176  001216  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
014324  001074                  BNE     15$             ;;BRANCH IF NO
014326  105777  177760          TSTB    @$TKS           ;;CHAR THERE?
014332  100071                  BPL     15$             ;;IF NO, DON'T WAIT AROUND
014334  117746  177754          MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
014340  042716  177600          BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
014344  022726  000007          CMP     #7,(SP)+        ;;IS IT A CONTROL G?
014350  001062                  BNE     15$             ;;NO, RETURN TO USER
014352  123727  015072  000001  CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
014360  001456                  BEQ     15$             ;;BRANCH IF YES

014362  104401  015043          TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
014366  104401  015050  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
014372  013746  000176          MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
014376  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
014400  104401  015061          TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
014404  005046          19$:    CLR     -(SP)           ;;CLEAR COUNTER
014406  005046                  CLR     -(SP)           ;;THE NEW SWR
014410  105777  177676  7$:     TSTB    @$TKS           ;;CHAR THERE?
014414  100375                  BPL     7$              ;;IF NOT TRY AGAIN

014416  117746  177672          MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
014422  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII


014426  021627  000025  9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
014432  001005                  BNE     10$             ;;BRANCH IF NOT
014434  104401  015036          TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
014440  062706  000006  20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
014444  000757                  BR      19$             ;;LET'S TRY IT AGAIN


014446  021627  000015  10$:    CMP     (SP),#15        ;;IS IT A <CR>?
014452  001022                  BNE     16$             ;;BRANCH IF NO
014454  005766  000004          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
014460  001403                  BEQ     11$             ;;BRANCH IF YES
014462  016677  000002  164526  MOV     2(SP),@SWR      ;;SAVE NEW SWR
014470  062706  000006  11$:    ADD     #6,SP           ;;CLEAR UP STACK
014474  104401  013765  14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
014500  123727  015073  000001  CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
014506  001003                  BNE     15$             ;;BRANCH IF NOT
014510  012777  000100  177574  MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
014516  000002          15$:    RTI                     ;;RETURN
```

```
                                16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
                                        CMP     (SP),#60        ;;CHAR < 0?
                                        BLT     18$             ;;BRANCH IF YES
                                        CMP     (SP),#67        ;;CHAR > 7?
                                        BGT     18$             ;;BRANCH IF YES
                                        BIC     #60,(SP)+       ;;STRIP-OFF ASCII
                                        TST     2(SP)           ;;IS THIS THE FIRST CHAR
                                        BEQ     17$             ;;BRANCH IF YES
                                        ASL     (SP)            ;;NO, SHIFT PRESENT
                                        ASL     (SP)            ;;    CHAR OVER TO MAKE
                                        ASL     (SP)            ;;    ROOM FOR NEW ONE.
                                17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
                                        BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
                                        BR      7$              ;;GET THE NEXT ONE
                                18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
                                        BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB

                                ;;******************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
014600  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
014602  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
014610  105777  177476          1$:     TSTB    @$TKS           ;;WAIT FOR
014614  100375                          BPL     1$              ;;A CHARACTER
014616  117766  177472  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
014624  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
014632  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
014640  001013                          BNE     3$              ;;BRANCH IF NO
014642  105777  177444          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
014646  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
014650  117746  177440                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
014654  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
014660  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
014664  001366                          BNE     2$              ;;IF NOT DISCARD IT
014666  000750                          BR      1$              ;;YES, RESUME
014670  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
014676  002407                          BLT     4$              ;;BRANCH IF YES
014700  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
014706  003003                          BGT     4$              ;;BRANCH IF YES
014710  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
014716  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;******************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

014720  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
```

```
                                1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
                                2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
                                        BLOS    4$              ;;BR IF YES
                                        RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
                                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
                                10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
                                        BNE     3$              ;;SKIP IF NOT
                                4$:     TYPE    ,$QUES          ;;TYPE A '?'
                                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
                                3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
                                        TYPE    ,9$
                                        CMPB    #15,(R3)+       ;;CHECK FOR RETURN
                                        BNE     2$              ;;LOOP IF NOT RETURN
                                        CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
                                        TYPE    ,$LF            ;;TYPE A LINE FEED
                                        MOV     (SP)+,R3        ;;RESTORE R3
                                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
                                        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
                                        MOV     #$TTYIN,4(SP)
                                        RTI                     ;;RETURN
                                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
                                        .BYTE   0               ;;TERMINATOR
                                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
                                $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
                                $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
                                $MSWR:  .ASCIZ  <15><12>/SWR = /
                                $MNEW:  .ASCIZ  /  NEW = /

                                $AUTOB: .BYTE   0               ;;AUTO MODE FLAG
                                $INTAG: .BYTE   0               ;;INTERRUPT MODE FLAG
```

```
                                .SBTTL  TRAP DECODER

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                ;*GO TO THAT ROUTINE.

015074  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
015076  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
015102  005740                          TST     -(R0)           ;;BACKUP BY 2
015104  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
015106  006300                          ASL     R0              ;;POSITION FOR INDEXING
015110  016000  015130                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
015114  000200                          RTS     R0              ;;GO TO ROUTINE


                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

015116  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
015120  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
015126  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.

                                ;       ROUTINE
                                ;       -------
015130  015116                  $TRPAD: .WORD   $TRAP2
015132  013534                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
015134  014014                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
015136  013770                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
015140  014030                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

015142  014366                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

015144  014316                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
015146  014600                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
015150  014720                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
015152  014216                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
015154  014254                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
015156  006544                          XSUBSCOPE       ;;CALL=SUBSCOPE TRAP+13(104413)
```

```
                        .SBTTL  POWER DOWN AND UP ROUTINES

                        ;;**********************************************************************
                        ;POWER DOWN ROUTINE
015160 012737 015324 000024 $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
015166 012737 000340 000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
015174 010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015176 010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015200 010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015202 010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015204 010446                       MOV     R4,-(SP)        ;;PUSH R4 ON STACK
015206 010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
015210 017746 164002                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
015214 010637 015330                MOV     SP,$SAVR6       ;;SAVE SP
015220 012737 015232 000024         MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
015226 000000                       HALT
015230 000776                       BR      .-2             ;;HANG UP

                        ;;**********************************************************************
                        ;POWER UP ROUTINE
015232 012737 015324 000024 $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
015240 013706 015330                MOV     $SAVR6,SP       ;;GET SP
015244 005037 015330                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
015250 005237 015330         1$:    INC     $SAVR6          ;;WAIT FOR THE INC
015254 001375                       BNE     1$              ;;OF  WORD
015256 012677 163734                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
015262 012605                       MOV     (SP)+,R5        ;;POP STACK INTO R5
015264 012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
015266 012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
015270 012602                       MOV     (SP)+,R2        ;;POP STACK INTO R2
015272 012601                       MOV     (SP)+,R1        ;;POP STACK INTO R1
015274 012600                       MOV     (SP)+,R0        ;;POP STACK INTO R0
015276 012737 015160 000024         MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
015304 012737 000340 000026         MOV     #340,@#PWRVEC+2 ;;PRIO:7
015312 104401                       TYPE                    ;REPORT THE POWER FAILURE
015314 015332               $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
015316 012716                       MOV     (PC)+,(SP)      ;;RESTART AT RESTART
015320 001350               $PWRAD: .WORD   RESTART         ;;RESTART ADDRESS
015322 000002                       RTI
015324 000000               $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
015326 000776                       BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
015330 000000               $SAVR6: 0                       ;;PUT THE SP HERE
015332 005015 047520 042527 $POWER: .ASCIZ  <15><12>"POWER"
015340 000122
                                    .EVEN
```

```
                                .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

                                ;;*********************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED DECIMAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
                                ;*      JSR     PC,@#$SB2D      ;;CALL
                                ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK

015342  016637  000002  015372 $SB2D:  MOV     2(SP),1$        ;;SAVE BINARY NUMBER
015350  012746  015372                  MOV     #1$,-(SP)       ;;SET POINTER
015354  004737  015376                  JSR     PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
015360  062716  000005                  ADD     #5,(SP)         ;;ONLY ALLOW FIVE CHARACTERS
015364  012666  000002                  MOV     (SP)+,2(SP)     ;;PICKUP POINTER
015370  000207                          RTS     PC              ;;RETURN
015372  000000  000000          1$:     .WORD   0,0
                                .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                                ;;*********************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
                                ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                                ;*POSITIVE.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#$DB2D
                                ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                                                ;;IS ON THE STACK

015376  104411          $DB2D:  SAVREG                          ;;SAVE REGISTERS
015400  016602  000002          MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
015404  012700  015556          MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
015410  010066  000002          MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
015414  012201                  MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
015416  012202                  MOV     (R2)+,R2
015420  012737  000012  015474  MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
015426  012704  015506          MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
015432  012705  015510          MOV     #$TNPWR+2,R5
015436  005003          1$:     CLR     R3              ;;CLEAR PARTIAL
015440  161401          2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
015442  005602                  SBC     R2
015444  161502                  SUB     (R5),R2
015446  002402                  BLT     3$              ;;BR IF TEN POWER TO LARGE
015450  005203                  INC     R3              ;;ADD 1 TO PARTIAL
015452  000772                  BR      2$              ;;LOOP
015454  062401          3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
015456  005502                  ADC     R2
015460  062402                  ADD     (R4)+,R2
015462  022525                  CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
015464  052703  000060          BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
015470  110320                  MOVB    R3,(R0)+        ;;SAVE IT
015472  005327                  DEC     (PC)+           ;;DONE?
015474  000000          4$:     .WORD   0
015476  001357                  BNE     1$              ;;BR IF NO
```

```
4066  015500  105020                    CLRB    (R0)+           ;;TERMINATOR
4067  015502  104412                    RESREG                  ;;RESTORE REGISTERS
4068  015504  000207                    RTS     PC              ;;RETURN
4069  015506  145000          $TNPWR:   145000                  ;;1.0E09
4070  015510  035632                    35632
4071  015512  160400                    160400                  ;;1.0E08
4072  015514  002765                    2765
4073  015516  113200                    113200                  ;;1.0E07
4074  015520  000230                    230
4075  015522  041100                    041100                  ;;1.0E06
4076  015524  000017                    17
4077  015526  103240                    103240                  ;;1.0E05
4078  015530  000001                    1
4079  015532  023420                    23420                   ;;1.0E04
4080  015534  000000                    0
4081  015536  001750                    1750                    ;;1.0E03
4082  015540  000000                    0
4083  015542  000144                    144                     ;;1.0E02
4084  015544  000000                    0
4085  015546  000012                    12                      ;;1.0E01
4086  015550  000000                    0
4087  015552  000001                    1                       ;;1.0E00
4088  015554  000000                    0
4089  015556  000014          $DECVL:   .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
4090
```

```
                        ;;*****************************************************************

                                ;TYPE NUMERICAL ASCIZ STRING,RIGHT JUSTIFIED
                                ;REPLACING LEADING ZEROS WITH SPACES.
                                ;
                                ;FIRST ADDRESS OF ASCIZ STRING MUST BE ON TOP OF THE STACK

015572  010046          RTJUST:         MOV R0,-(SP)            ;SAVE R0
015574  016600  000004                  MOV 4(SP),R0            ;PICK UP ADDRESS OF ASCIZ STRING
015600  010037  015632                  MOV R0,3$               ;SAVE ADDRESS FOR TYPE OUT
015604  105710          1$:             TSTB (R0)               ;IS THIS THE TERMINATOR
015606  001406                          BEQ 2$                  ;IF YES TYPE IT OUT
015610  122710  000060                  CMPB #'0,(R0)           ;IS IT A ZERO
015614  001005                          BNE 4$                  ;IF NO GO PRINT IT
015616  112720  000040                  MOVB #' ,(R0)+          ;IF YES REPLACE IT WITH A SPACE
015622  000770                          BR 1$                   ;TEST NEXT CHAR.
015624  112740  000060          2$:     MOVB #'0,-(R0)          ;STRING OFF ALL ZEROS,PUT BACK THE LAST ONE
015630  104401          4$:             TYPE                    ;TYPE THE STRING
015632  000000          3$:             OPEN
015634  012600                          MOV (SP)+,R0            ;RESTORE R0
015636  012616                          MOV (SP)+,(SP)          ;RESTORE THE STACK
015640  000207                          RTS PC                  ;RETURN

                                ;TYPES 16 BIT WORD IN DECIMAL

015642  012546          SGLDEC:         MOV (R5)+,-(SP)         ;PUT NUMBER TO BE TYPED ON STACK
015644  004737  015342                  JSR PC,@#$SB2D          ;CONVERT NUMBER TO DECIMAL
015650  004737  015572                  JSR PC,RTJUST           ;TYPE THE DECIMAL NUMBER
015654  000205                          RTS R5
```

```
                                        .SBTTL  MESSAGES

015656  042524  052123  050040  MDTESTP:        .ASCIZ "TEST PARAMETERS: "
015664  051101  046501  052105
015672  051105  035123  000040

015700  005015  042412  040522  MXEHEADER:      .ASCIZ <15><12><12> "ERADR  FAST   FAPT          GOOD   BAD     PASS"
015706  051104  020040  040506
015714  052123  020040  043040
015722  050101  020124  020040
015730  020040  020040  020040
015736  043440  047517  020104
015744  020040  040502  020104
015752  020040  020040  040520
015760  051523     000
015763     125  042516  050130  MUNXDD:         .ASCIZ "UNEXPECTED D D MARK"<15><12>
015770  041505  042524  020104
015776  020104  020104  040515
016004  045522  005015     000

016011     104  042040  046440  MDDMIS:         .ASCIZ "D D MARK MISSING"<15><12>
016016  051101  020113  044515
016024  051523  047111  006507
016032  000012

016034  040504  040524  020054  MDERHDR:        .ASCIZ "DATA, NO STATUS ERROR"
016042  047516  051440  040524
016050  052524  020123  051105
016056  047522  000122

016062  047440  020116  051124  MTRK:           .ASCIZ " ON TRACK"
016070  041501  000113

016074  020040  051106  046517  MPRES:          .ASCIZ "  FROM TRACK"
016102  052040  040522  045503
016110     000

016111     040  020057  042523  MSECT:          .ASCIZ " / SECTOR"
016116  052103  051117     000

016123     015  020012  054502  MCOLMUN:        .ASCIZ <15><12>" BYTE  BAD  GOOD"<15><12>
016130  042524  020040  040502
016136  020104  043440  047517
016144  006504  000012

016150  020040  020040  020040  TAB:            .ASCIZ <40><40><40><40><40><40>
016156     000

016157     040  000040          DBLSP:          .ASCIZ <40><40>

016162  005015     000          MCRLF:          .ASCIZ <15><12>

016165     015  042412  051122  MERHEADER:      .ASCIZ <15><12>"ERROR CONDITIONS:  TEST PC = "
016172  051117  041440  047117
```

```
 4176  016200  044504  044524  047117
 4177  016206  035123  020040  042524
 4178  016214  052123  050040  020103
 4179  016222  020075     000
 4180
 4181  016225     125  044516  020124  MUNIT0:         .ASCIZ "UNIT 0 "
 4182  016232  020060     000
 4183
 4184  016235     125  044516  020124  MUNIT1:         .ASCIZ "UNIT 1 "
 4185  016242  020061     000
 4186
 4187  016245     117  046116  020131  MONLY:          .ASCIZ "ONLY  "
 4188  016252  000040
 4189
 4190  016254  054122  051503  036440  MRXCS:          .ASCIZ "RXCS = "
 4191  016262  000040
 4192
 4193  016264  052123  052101  051525  MASTAT:         .ASCIZ "STATUS A = "
 4194  016272  040440  036440  000040
 4195
 4196  016300  052123  052101  051525  MBSTAT:         .ASCIZ "STATUS B = "
 4197  016306  041040  036440  000040
 4198
 4199  016314  005015  000012          DBLLF:          .ASCIZ <15><12><12>
 4200
 4201  016320  047516  044440  052116  MINTER:         .ASCIZ "NO INTERRUPT AT DONE ERROR"
 4202  016326  051105  052522  052120
 4203  016334  040440  020124  047504
 4204  016342  042516  042440  051122
 4205  016350  051117     000
 4206
 4207  016353     125  045516  047516  MUKNINT:        .ASCIZ "UNKNOWN INTERRUPT"
 4208  016360  047127  044440  052116
 4209  016366  051105  052522  052120
 4210  016374     000
 4211
 4212  016375     124  052117  046101  MERCT:          .ASCIZ "TOTAL READ CHECK ERRORS = "
 4213  016402  051040  040505  020104
 4214  016410  044103  041505  020113
 4215  016416  051105  047522  051522
 4216  016424  036440  000040
 4217
 4218  016430  044506  046114  052502  MFIL:           .ASCIZ "FILLBUFFER "
 4219  016436  043106  051105  000040
 4220
 4221  016444  046505  052120  041131  MEMPTY:         .ASCIZ "EMPTYBUFFER "
 4222  016452  043125  042506  020122
 4223  016460     C00
 4224
 4225  016461     040  051124  041501  MLIMTRK:        .ASCIZ " TRACKS 52,53,114,0  "
 4226  016466  051513  032440  026062
 4227  016474  031465  030454  032061
 4228  016502  030054  020040     000
 4229
 4230  016507     117  036504     000  MOD:            .ASCIZ "OD="
 4231
```

```
 4232  016513     040  044440  036504  MID:        .ASCIZ "  ID="
 4233  016520     000
 4234
 4235  016521     040  043040  051111  MFIRST:     .ASCIZ "  FIRST="
 4236  016526  052123  000075
 4237
 4238  016532  020040  040514  052123  MLAST:      .ASCIZ "  LAST="
 4239  016540  000075
 4240
 4241  016542  051103  020103  051105  MBADCRC:    .ASCIZ "CRC ERROR NO DATA ERROR"
 4242  016550  047522  020122  047516
 4243  016556  042040  052101  020101
 4244  016564  051105  047522  000122
 4245
 4246  016572  042522  042101  000040  MREAD:      .ASCIZ "READ "
 4247
 4248  016600  040504  040524  041440  MCRC:       .ASCIZ "DATA CRC ERROR"
 4249  016606  041522  042440  051122
 4250  016614  051117     000
 4251
 4252  016617     123  042505  020113  MSEEK:      .ASCIZ "SEEK ERROR"
 4253  016624  051105  047522  000122
 4254
 4255  016632  051127  052111  020105  MWRITE:     .ASCIZ "WRITE "
 4256  016640     000
 4257
 4258  016641     120  051101  052111  MPAR:       .ASCIZ "PARITY ERROR"
 4259  016646  020131  051105  047522
 4260  016654  000122
 4261
 4262  016656  051105  047522  020122  MNOFLAG:    .ASCIZ "ERROR FLAG ERROR"
 4263  016664  046106  043501  042440
 4264  016672  051122  051117     000
 4265
 4266  016677     102  042101     000  MBAD:       .ASCIZ "BAD"
 4267
 4268  016703     040     000          SPACE:      .ASCIZ <40>
 4269
 4270  016705     107  047517  000104  MGOOD:      .ASCIZ "GOOD"
 4271
 4272  016712  020040  044103  041505  MSUM:       .ASCIZ "  CHECK SUM "
 4273  016720  020113  052523  020115
 4274  016726     000
 4275
 4276  016727     015  051012  030530  MRX11:      .ASCIZ <15><12>"RX11 / RXV11"
 4277  016734  020061  020057  054122
 4278  016742  030526  000061
 4279
 4280  016746  005015  041412  051132  MREV:       .ASCIZ <15><12><12> "CZRXBF0 RX11 INTERFACE TEST" <15><12>
 4281  016754  041130  030106  051040
 4282  016762  030530  020061  047111
 4283  016770  042524  043122  041501
 4284  016776  020105  042524  052123
 4285  017004  005015     000
 4286
 4287  017007     015  052412  042516  LOC4M:  .ASCIZ  <15><12>"UNEXPECTED TRAP TO LOC. 4 OCCURRED"
```

```
 4288  017014  050130  041505  042524
 4289  017022  020104  051124  050101
 4290  017030  052040  020117  047514
 4291  017036  027103  032040  047440
 4292  017044  041503  051125  042522
 4293  017052  000104
 4294
 4295  017054  005015  047125  054105  LOC10M: .ASCIZ  <15><12>"UNEXPECTED TRAP TO LOC. 10 OCCURRED"
 4296  017062  042520  052103  042105
 4297  017070  052040  040522  020120
 4298  017076  047524  046040  041517
 4299  017104  020056  030061  047440
 4300  017112  041503  051125  042522
 4301  017120  000104
 4302
 4303  017122  050075  000103          PCM:    .ASCIZ  "=PC"
 4304
 4305  017126  005015  051124  041501  OD2BIG: .ASCII  <15><12>"TRACK LIMITS SELECTED OUT OF RANGE"
 4306  017134  020113  044514  044515
 4307  017142  051524  051440  046105
 4308  017150  041505  042524  020104
 4309  017156  052517  020124  043117
 4310  017164  051040  047101  042507
 4311  017172  005015  042504  040506          .ASCIZ  <15><12>"DEFAULTING TO "
 4312  017200  046125  044524  043516
 4313  017206  052040  020117     000
 4314
 4315  017213     015  051412  041505  S2BIG:  .ASCII  <15><12>"SECTOR LIMITS SELECTED OUT OF RANGE"
 4316  017220  047524  020122  044514
 4317  017226  044515  051524  051440
 4318  017234  046105  041505  042524
 4319  017242  020104  052517  020124
 4320  017250  043117  051040  047101
 4321  017256  042507
 4322  017260  005015  042504  040506          .ASCIZ  <15><12>"DEFAULTING TO "
 4323  017266  046125  044524  043516
 4324  017274  052040  020117     000
 4325
 4326  017301     015  041412  052501  DOLOAD: .ASCII  <15><12>"CAUTION - IF YOU DESIRE TO TEST UNIT 0"
 4327  017306  044524  047117  026440
 4328  017314  044440  020106  047531
 4329  017322  020125  042504  044523
 4330  017330  042522  052040  020117
 4331  017336  042524  052123  052440
 4332  017344  044516  020124     060
 4333  017351     015  051012  050105          .ASCII  <15><12>"REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE"
 4334  017356  040514  042503  046040
 4335  017364  040517  020104  042515
 4336  017372  044504  046525  053440
 4337  017400  052111  020110  020101
 4338  017406  041523  040522  041524
 4339  017414  020110  044504  045523
 4340  017422  052105  042524
 4341  017426  005015  044124  047105          .ASCIZ  <15><12>"THEN PRESS CONTINUE"<15><12>
 4342  017434  050040  042522  051523
 4343  017442  041440  047117  044524
```

```
 4344  017450  052516  006505  000012
 4345
 4346                                  .EVEN
 4347
 4348                                  ;;*****************************************************************
 4349
 4350                                  ;THE FOLLOWING LOCATIONS ARE USED FOR DATA STORAGE,RETRY COUNTERS
 4351                                  ;ACCESS COUNTERS ETC.
 4352
 4353  017456  000200                  BUFADR:         .BLKB 200
 4354
 4355          000001                  .END
```

CROSS REFERENCE TABLE -- USER SYMBOLS

```
ADJSUM  011014          2322    2395    2413    2751    3030    3048#
ASTAT   012174          2808    2917    2924    2973    2991    3159*   3162    3165    3242#   3246*   3256
BADBYT  011502          3041*   3042    3043    3089    3137#
BIT0  = 000001           176#   2924
BIT00 = 000001           166#    176
BIT01 = 000002           165#    175
BIT02 = 000004           164#    174
BIT03 = 000010           163#    173
BIT04 = 000020           162#    172
BIT05 = 000040           161#    171
BIT06 = 000100           160#    170
BIT07 = 000200           159#    169
BIT08 = 000400           158#    168
BIT09 = 001000           157#    167
BIT1  = 000002           175#   2808    2917
BIT10 = 002000           156#
BIT11 = 004000           155#
BIT12 = 010000           154#
BIT13 = 020000           153#   2873
BIT14 = 040000           152#    450    1432    3433    3447
BIT15 = 100000           151#    455     461     469    3022    3467    3505
BIT2  = 000004           174#   3162
BIT3  = 000010           173#
BIT4  = 000020           172#   3234    3573
BIT5  = 000040           171#   1540    2637
BIT6  = 000100           170#    669     829    1083    1244    1288    1302    1342    1360    1402    1421    1489    1593
                        1832    1851    1957    2973    2991    3424    3429    3435
BIT7  = 000200           169#    672     673    1383    1492    1493    1597    2448    2454    3029    3084    3510    3575
BIT8  = 000400           168#    547     831     865    2529    2561    2980    2993    3068    3184
BIT9  = 001000           167#
BPTVEC= 000014           183#
BRLEV   001214           371#   2492
BSTAT   012176          3192*   3243#   3254*   3262
BUFADR  017456          1625    1728    3048*   3049*   3053*   3056*   3057    3277    3379    4353#
BUSERR  006156           211    2460#
BYTECN  011500          2754*   2762*   3027*   3045*   3085    3102*   3136#
CATERR  003060          1171    1176*   1179#   1191*
CCOUNT  002554           439*    834     837*    838     840*    878#   2528*
CEXIT   004274          1066    1304    1361    1422    1608    1618    1725    1811    1812#
CKBYTE  010764          3041#
CKSUM   011104          3031*   3042*   3050*   3051*   3052*   3053    3054*   3055*   3056    3060#   3118
CKSWR = 104406           845     892    2525    2566    2572    2622    2671    2771    2777    2826    2831    2836    2864
                        2872    2876    2939    2944    2955    2960    2975    2983    2994    3070    3098    3110    3124
                        3145    3151    3169    3177    3193    3959#
COMMWO  007662          2333    2793    2851#   2896
CONT10  010550          2972    2991#
CONT11  010516          2977    2983#   2996
CONT13  007340          2779    2781#
CONT14  007610          2833    2835#
CONT15  010340          2946    2948#
CONT16  010412          2962    2964#
CONT17  010530          2985    2987#
CONT20  011324          3100    3102#
CONT21  011550          3153    3155#
CONT4   007326          2773    2777#
COUNT   012476          3288    3363#
```

```
CPUPRI  006226          1335    1394    2492#
CR    = 000015            91#   3637    3647
CRCER   010262          2925    2933#
CRLF  = 000200            92#   3608    3647
DATABY  012350          3292#   3301*   3311*   3317*   3327*   3335*   3344*   3346*   3355*   3363*   3364*   3373*   3377
                        3378
DATACK  007732          2862#   2891*   2935*   3105
DATACR  010356          2934    2938    2955#
DATAER  011136          3044    3068#
DATAO   012354          3282    3301#
DATA1   012366          3283    3311#
DBLLF   016314          4199#
DBLSP   016157          3088    3092    3233    4170#
DDCHK   010436          2908    2971#
DDERR   010500          2980#   2999
DDISP = 177570            98#    379
DEATH   003062          1175    1185#
DEXIT   004306          1815#   1897    2031    2085    2137    2272    2309    2332    2400    2416    2429    2455
DING    006476          2562    2572#   3013
DISPLA  001220           379#    420*
DISPRE  000174           238#    420
DONE    012754          2744    3445#
DONEBI= 000040           198#   2797    2900    3167
DRVSWP  007040          2428    2718#
DSWR  = 177570            97#    378
DTESTP  001212           369#    446     450     453     472     866
DOLOAD  017301          4326#
EBIE  = 000103           200#   3037
EMPBUF  010672          2936    3024#   3064    3128    3132
EMPDON  011334          3032    3105#
EMPER   011106          3033    3062#
EMPFLA  010756          3038#   3039    3046    3103
EMTVEC= 000030           186#
EPCSCO  006522          2518*   2519*   2582#   2614
ERCNTR  011506          2937    3028*   3069*   3075    3108    3114    3139#
ERMSG   010614          2978    2997    3003#   3079    3148    3197
ERRORS  006520           548*   2520*   2581#   2587    2596*   2602    3182*
ERRVEC= 000004           179#
EXIT    011474          3026    3130    3133#
FAST    002560           550*    757*    880#   2541
FBEB    004112          1626    1635#   1641    1729    2325
FBIE  = 000101           199#   2758
FILLBU  007172          2751#   2768    2783    2817    2840
FILLDO  007360          2752    2786#   2813
FILLER  007250          2753    2765#
FILLFL  007230          2759#   2760    2763
FIRST   001202           302#    303     508     510*    513     516     530*   3548    3550*   3553    3556    3570
FIRSTT  002206           756#    873
FLOAT0  012376          3284    3317#
FLOAT1  012440          3285    3335#
FUNCTI  007730          2330*   2414*   2789*   2792*   2851*   2852    2861#   2895*   3199
GETPAT  012306          1624    2321    2394    2412    2718    2736    3277#
GETSEC  013450          2750    2890    3566#
GETTRA  013132          2707    2721    2739    3486#
GETUNI  012662          2319    2411    2706    2720    2724    2738    3424#
GNS   = ****** U        3952    3953    3954    3955    3957    3959    3960   3961    3962    3963    3964
```

```
GOODBY  011504            3093*  3095   3138#
GTSWR = 104405             426   3957#
HANGER  007004             441*   555*   830*   854*  2639*  2669*  2694*  2698#  2786*  2799*  2892*  2902*
HANGPL  007006             442*   556*   665*  2640*  2670*  2695*  2699#
HERE    002500             857    863#
HLT10   010336            2947#
HLT12   010410            2963#
HLT13   010526            2986#
HLT14   011322            3101#
HLT15   011546            3154#
HLT6    007336            2780#
HLT7    007606            2834#
HOME    007734            2835   2864#  2881
HT    = 000011             89#   3606   3647
HUNGUP  006650            2641   2648#
ID    = 001201             301#   485    487    495   3470
ILLADR  005134            1914   2047   2100   2141#  2146
INCERR  006272            2520#  2521
INITSE  013352            2749   2889   3548#
INITTR  013000            2705   2719   2737   3462#
INTLEA  013446            3559*  3564#  3571   3572*
INTSER  011554            2306   2426   2451   3155   3159#
IOTVEC= 000020             184#
KRXVEC  001204             305#  1019   1073   1237   1278   1320   1376   2305   2425   2450
LAST  = 001203             303#   511    513    521   3551*  3552   3568
LF    = 000012              90#  3641   3647
LIMITS  001640             477#   504
LIMTRK  013166            3488   3502#
LOAD    012536            3302   3322   3345   3365   3374   3377#
LOCKUP  002604             754    892#
LOC10M  017054            2476   4295#
LOC4M   017007            2460   4287#
LOGICA  002470             231    859#
MABELL  006516             842   2575   2578#
MASTAT  016264            3255   4193#
MBAD    016677            3120   4266#
MBADCR  016542            2942   4241#
MBSTAT  016300            3261   4196#
MCOLMU  016123            3083   4162#
MCRC    016600            2958   4248#
MCRLF   016162             476    525    836    848   2533   2776   2848   2943   2959   2982   3012   3081   3097
                          3123   3150   3173   3181   3196   3204   3218   3264   4172#
MDDMIS  016011            2979   4141#
MDERHD  016034            3080   4147#
MDTEST  015656             471   4123#
MEMPTY  016444            3063   4221#
MERCT   016375            3113   4212#
MERHEA  016165            3003   4174#
MFIL    016430            2766   4218#
MFIRST  016521             515   4235#
MGOOD   016705            3122   4270#
MHUNGP  006722            2675   2685#
MID     016513             494   4232#
MILTST  002564             875    884#
MINTER  016320            3149   4201#
MLAST   016532             520   4238#
```

```
MLIMTR  016461            480   4225#
MNOFLA  016656           3180   4262#
MOD     016507            489   4230#
MONLY   016245            467   4187#
MORETE  002200            754#  1813   1816   2674   2683
MPAR    016641           2775   3217   4258#
MPASS   006753            849   2690#  3008
MPRES   016074           2843   4155#
MREAD   016572           2920   2928   4246#
MREV    016746            412   4280#
MRXCS   016254           3198   3212   4190#
MRX11   016727           3209   4276#
MSECT   016111           3226   4159#
MSEEK   016617           2842   4252#
MSUM    016712           3117   4272#
MTRK    016062           3221   4152#
MUKNIN  016353           3172   4207#
MUNITO  016225           3236   4181#
MUNIT1  016235            466   3238   4184#
MUNXDD  015763           2998   4136#
MWRITE  016632           2811   2822   2829   4255#
MX      002562            833*   841    882#  1812*  1815*
MXEHEA  015700           2531   4127#
NEXTRD  010160           2784   2907   2910   2912#  2922   2930   2950   2967   2990
NOERTY  011312           3072   3076   3098#
NOINTE  011510           2801   2904   3145#
NOMORE  002312            766    829#
NOPRIN  006446           2527   2561#
NOPRNT  011754           3193#
NOSCOP  006532           2596#  2624
NXREAD  011456           3065   3109   3126   3129#
OD      001200            300#   301    470*   477    487    490    503*  3464*  3465   3468   3472   3476*  3502
                         3505*  3507   3510*  3524*  3527   3537   3539*
OD2BIG  017126            500   4305#
OK2GO   003064           1178   1190#
OPEN  = 000000            205#   852   2558   2774   3011   3087   3116   3240   4109
PARTES  007300           2769   2771#  2814   2923   3066
PARTYP  012050           3190   3212#  3252
PASS    002556            440*   843*   850    879#  2556   3009
PAT     012352           1616*  1803*  1808*  2320*  2391*  2393*  2408*  2410*  2424*  2437*  2449*  3279   3293#
PATGEN  012360           3302#  3303   3312
PATTER  012330           3281   3282#
PAT125  012446           3286   3344#
PAT314  012466           3287   3355#
PCM     017122           2467   2484   4303#
PCONT   007354           2767*  2782   2784#  2812*  2922*  3065*
PCSCOP  006604            550    756*   757   2544   2597*  2625   2627#  2790   2906   2971   3004
PIRQ  = 177772             96#
PIRQVE= 000240            190#
PLOOP   007350           2768*  2783#  2813*  2921*  3064*
PRESTR  013156           2844   2883*  3486*  3494#  3525
PRETES  002052            526    547#
PR0   = 000000            113#
PR1   = 000040            114#
PR2   = 000100            115#
PR3   = 000140            116#
```

```
PR4    = 000200   117#  1329
PR5    = 000240   118#  1388
PR6    = 000300   119#
PR7    = 000340   120#   212   214   218   220   222   224   432  1023  1031  1077  1126  1241
                 1264  1282  1291  1324  1345  1380  1405  2307  2427  2452
PS     = 177776    93#    94
PSW    = 177776    94#  1329*  1388*
PTYP1    007312  2766*  2774#  2811*  2920*  3063*
PWRVEC= 000024   185#  3969*  3970*  3979*  3985*  3997*  3998*
RANDAT   012516  3289  3372#  3375
RANGEN   012566  3372  3387#
RANUM    012660  3373  3402*  3407#
RAN1     012654   437*  3388  3394*  3399  3405#
RAN2     012656   438*  3389  3396  3401*  3406#
RDCHR  = 104407  3893  3960#
RDCODE   012200  2875  3246#
RDDONE   010136  2893  2906#
RDER   = 000017   197#  3185  3247
RDERR    010170  2894  2916#
RDIE   = 000107   202#  2895
RDLIN  = 104410  3961#
RDONLY   007010  2308  2705#
READ     010042  2708  2889#  3023
READCH   010662  2723  2726  2741  3022#
REBEGI   002504   675   753   863   865#  2468  2485
RECAL  = 040001   204#  2396  2867
REREAD   010052  2891#  2921  2927  2951  2968  2989  3127
RESERR   006202   213  2476#
RESREG= 104412  3963#  4067
RESTAR   001350   405   431#  3211  4002
RESVEC= 000010   180#
RETURN   010612  2974  2992  3000#
RTJUST   015572  4098#  4118
RTN      010040  2866  2884#
RTSPC    006620  2631  2633#
RXCS     001206   307#   435   552   829*  871   918*  920  1083*  1197  1199  1244*  1249  1251
                1288*  1302*  1342*  1360*  1402*  1421*  1432*  1436  1539*  1540  1542  1552  1623*
                1642  1727*  1826*  1845*  1911*  1913*  1917  1969*  1972  1975  2044*  2046*  2050
                2074*  2097*  2099*  2103  2126*  2166  2176*  2182  2191  2198  2249*  2252  2255
                2324*  2336  2339*  2396*  2630  2637  2758*  2759  2797  2852*  2867*  2870  2900
                3037*  3038  3160  3167  3185*  3213  3247*
RXDB     001210   349#   435*   436*   667   974  1487  1591  1629*  1644  1735  1830  1849  1955
                1976  1986  2064  2078  2116  2130  2148*  2156*  2183  2185*  2192  2194*  2238
                2256  2264  2327*  2338  2342  2761*  2855*  2858*  3041  3159  3188  3192  3246
                3250  3254
RXERRO   011700  3161  3179  3182#  3183
RXPWR    012034  3164  3209#
SAVREG= 104411  3962#  4041
SA200    001232   245   409#   876
SA202    001222   246   403#
SCOPIN   006564  2588  2615  2622#
SDN      006622  1433  1549  1640  1827  1846  1970  2075  2127  2145  2153  2162  2180  2189
                2195  2250  2334  2340  2397  2637#  2868  3186  3248
SECCNT   013442  2802*  2912*  3024  3129*  3552*  3553*  3554*  3555*  3562#
SECLMT   001744   481   499   508#
SEEKER   007554  2823  2826#  2929
```

```
SEKRTY  007630  2817* 2838  2840# 2927*
SEKTYP  007634  2830  2842#
SEQUEN  013164   479* 2448* 2454* 3462  3467* 3487  3497#
SEQ1    013264  3489  3524#
SEQ2    013320  3490  3537#
SGLDEC  015642   851  2557  3010  3086  3115  4116#
SPACE   016703  2540  2543  2546  2549  2552  2555  4268#
STACK = 001200    84#  298   404   411   758
STKLMT= 177774    95#
STR     006606  1635  2141  2149  2157  2177  2186  2630#  2853  2856
STYP1   007566  2822* 2829# 2928*
SUBSCO= 104413   958  1016  1190  1229  1256  1274  1471  1533  1548  1586  1673  1839  1949
                1964  1981  2057  2070  2110  2122  2231  2245  2261  2376  3964#
SUM     012564  3050  3278* 3377* 3385#
SWHLT1  007576  2828  2831#
SWR     001216   378#  410*  415   419*  424   460*  461*  846   893   899   2526  2567  2573
                2623  2672  2772  2778  2781  2827  2832  2837  2865  2873  2877  2880  2940
                2945  2948  2956  2961  2964  2976  2984  2987  2995  3071  3073  3099  3111
                3125  3146  3152  3170  3178  3194  3790  3827* 3977  3990*
SWREG   000176   239#  419   424   460   3790  3803
SW0   = 000001   148#  2573
SW00  = 000001   138#  148
SW01  = 000002   137#  147
SW02  = 000004   136#  146
SW03  = 000010   135#  145
SW04  = 000020   134#  144
SW05  = 000040   133#  143
SW06  = 000100   132#  142
SW07  = 000200   131#  141
SW08  = 000400   130#  140
SW09  = 001000   129#  139
SW1   = 000002   147#
SW10  = 002000   128#  893
SW11  = 004000   127#  2623  2781  2837  2880  2948  2964  2987  3125
SW12  = 010000   126#  899
SW13  = 020000   125#  2526  2672  2772  2827  2940  2956  2976  2995  3071  3111  3146  3170
                3178  3194
SW14  = 040000   124#  846
SW15  = 100000   123#  2567
SW2   = 000004   146#
SW3   = 000010   145#
SW4   = 000020   144#
SW5   = 000040   143#
SW6   = 000100   142#
SW7   = 000200   141#
SW8   = 000400   140#  2865
SW9   = 001000   139#  3073
S2BIG   017213   527   4315#
TAB     016150  3260  4167#
TARGET  013154  2317* 2858  3048  3051  3222  3468* 3486  3493# 3504* 3509* 3512  3514* 3516*
                3527* 3529* 3540* 3542*
TBITVE= 000014   181#
TESTS   002232   756   759   760#
TKVEC = 000060   188#
TPVEC = 000064   189#
TRAPVE= 000034   187#
```

```
TRKCNT  013152  2709* 2727* 2742* 3473* 3474* 3475* 3478* 3480* 3492#
TRKLMT  001660   478  485#
TRTVEC= 000014   182#
TSECTO  013444  2318* 2855  3049  3052  3227  3556* 3557* 3563# 3566* 3567* 3568  3570* 3571*
                3575*
TYPADR  012076  2981  3082  3203  3221#
TYPCOD  012246  3205  3255#
TYPE  = 104401   412   466   467   471   476   480   489   494   500   515   520   525   527
                 836   841   842   848   849   875  2460  2467  2476  2484  2531  2533  2540
                2543  2546  2549  2552  2555  2575  2675  2774  2775  2776  2829  2842  2843
                2848  2942  2943  2958  2959  2979  2982  2998  3003  3008  3012  3080  3081
                3083  3088  3092  3097  3113  3117  3120  3122  3123  3149  3150  3172  3173
                3180  3181  3196  3198  3204  3209  3210  3212  3217  3218  3221  3226  3233
                3236  3238  3255  3260  3261  3264  3611  3716  3801  3802  3805  3818  3829
                3848  3897  3900  3904  3952#  3999  4108
TYPERR  011236  3078  3085#
TYPOC = 104402  3804  3953#
TYPON = 104404  2542  2545  2548  2551  2554  3096  3263  3955#
TYPOS = 104403   473   491   496   517   522  2464  2481  2537  2680  2845  3005  3090  3200
                3214  3223  3230  3257  3954#
TONOTH  002204   755#  760
T1      002646   760   918#
T10     004166   763  1716#
T11     004254   763  1808#
T12     004244   763  1803#
T13     004320   763  1824#
T14     004444   763  1907#
T15     004636   763  2040#
T16     004776   763  2093#
T17     005244   763  2174#
T2      002766   760  1073#
T20     005510   766  2282#  2906
T21     005534   766  2317#
T21X    005636  2331  2333#  2415
T22     005720   766  2389#
T23     005776   766  2408#
T24     006044   766  2424#
T24X    006052  2425#  2438
T25     006076   766  2437#  2790  2971
T26     006110   766  2448#
T3      003310   760  1317#
T4      003446   760  1373#
T5      003612   760  1432#
T6      004036   760  1616#
T6FILL  004054  1617  1622#  1809  2392  2409
T7      004170   760  1724#
T7EMPT  004200  1724  1726#  1810  2379  2399
UNITSE  012752   445*  448*  452*  455*  465*  469*  547*  670   831   865*  903*  1490  1595
                1824  1843  1909  2042  2095  2389  2529  2561* 2851  2980* 2993* 3068* 3184*
                3234  3424  3426  3428* 3429* 3431  3433  3435* 3436* 3441# 3445  3447  3573
WRITE   007162  2722  2725  2740  2749#
WRTDON  007456  2787  2802#  2812  2824
WRTER   007472  2788  2807#
WRTIE = 000105   201#  2789
WRTSEK  007532  2809  2817#
WTDDIE= 000115   203#  2792
```

```
WTRDCK  007114            2453    2736#
XERROR  006260             221    2518#
XFRBYT  007236            2761#
XHOME   010032            2871    2883#
XID     013162            3469*   3470*   3473    3496#   3525
XOD     013160            3471*   3472*   3474    3495#
XPATGE  012404            3318#   3336
XREAD   010046            2890#   2913    3131
XSA202  001612             444     457     462     464     468     470#
XSCOPE  006524             217    2587#
XSDN    006764            2638    2694#
XSUBSC  006544            2602#   3964
XWRITE  007166            2750#   2767    2805
XWTRDC  007120            2737#   2746
XXPATG  012454            3345#   3347    3356
$AUTOB  015072             428*   3798    3919#
$CHARC  013750            3613*   3623*   3630    3639*   3644#
$CKSWR  014316            3790#   3959
$CNTLG  015043            3801    3914#
$CNTLU  015036            3818    3913#
$CRLF   013765            3612    3654#   3829    3913
$DB2D   015376            4023    4041#
$DECVL  015556            4043    4089#
$FILLC  013762            3616    3651#
$FILLS  013761            3650#
$GTSWR  014366            3802#   3957
$HD   = 000003              18      19
$ILLUP  015324            3969    3985    4004#
$INTAG  015073            3830    3920#
$LF     013766            3655#   3904    3913
$MAIL = ****** U           424    3600
$MNEW   015061            3805    3917#
$MSWR   015050            3802    3915#
$NULL   013760            3618    3649#
$OCNT   014212            3688*   3717*   3730#
$OMODE  014214            3683*   3687*   3692    3695*   3706*   3732#
$POWER  015332            3210    4000    4007#
$PWRAD  015320            4002#
$PWRDN  015160             219    3969#   3997
$PWRMG  015314            4000#
$PWRUP  015232            3979    3985#
$QUES   013764            3653#   3848    3897    3913
$RDCHR  014600            3861#   3960
$RDDEC= ****** U          3962
$RDLIN  014720            3889#   3961
$RDOCT= ****** U          3962
$RDSZ = 000010            3882#
$RESRE  014254            3766#   3963
$R2A  = ****** U          3964
$SAVRE  014216            3750#   3962
$SAVR6  015330            3978*   3986    3987*   3988*   4006#
$SB2D   015342            4021#   4117
$SETUP= 000114             241#    421    3785    3919
$STUP = 177777             241#
$SWR  = 160000              18     19#    4003
$TKB    014314            3782#   3794    3811    3865    3871
```

```
$TKS     014312          3781#  3792   3808   3832*  3863   3869
$TN    = 000001            18#
$TNPWR   015506          4048   4049   4069#
$TPB     013756          3636*  3648#
$TPFLG   013763          3594   3652#
$TPS     013754          3634   3647#
$TRAP    015074           223   3929#
$TRAP2   015116          3940#  3951
$TRP   = 000014          3944#  3953#  3954#  3955#  3956#  3957   3958#  3959   3960#  3961#  3962#  3963#  3964#
                         3965#
$TRPAD   015130          3934   3951#
$TTYIN   015026          3890   3891   3908   3912#
$TYPBN=  ******  U       3956
$TYPDS=  ******  U       3956
$TYPE    013534          3594#  3944   3952
$TYPEC   013704          3615   3622   3629   3634#  3635   3834
$TYPEX   013752          3640   3642   3645#
$TYPOC   014014          3686#  3953
$TYPON   014030          3685   3688#  3955
$TYPOS   013770          3681#  3954
$OFILL   014213          3682*  3686*  3696   3731#
.      = 017656           207#   209    210#   216#   227#   230#   234#   237#   242#   298#   888#   1804   2692#
                         3647   3648   3649   3650   3651   3652   3653   3654   3655   3656   3781   3782   3912#
                         3913   3919   3920   3921   3981   4005   4089#  4353#
```

```
COMMEN       1#      5#    191#    563     684     928     982    1042    1086    1134    1205    1444    1503    1560    1650
          1684    1748    1859    1925    1995    2208    2284    2348    2649
ENDCOM       1#      5#    191#    648     749     954    1012    1061    1121    1164    1225    1467    1529    1582    1669
          1703    1792    1892    1945    2026    2227    2301    2372    2664
ERROR       85#    561     681     926     980    1040    1132    1203    1255    1273    1297    1352    1415    1442    1501
          1547    1558    1606    1648    1682    1746    1838    1857    1923    1963    1980    1993    2056    2069    2083
          2109    2121    2135    2206    2244    2260    2270    2346    2648
ESCAPE       1#    191#
GETPRI       1#    191#
GETSWR       1#      5#    191#    421
MULT         1#    191#
M7727      249#    617     730    1448    1773    1863    1999    2352
M7846      247#    562     683     927     981    1041    1085    1133    1204    1502    1559    1649    1683    1747    1924
          2207
NEWTST       1#    191#
POP          1#    191#   3771    3990    3991
PUSH         1#    191#   3751    3971    3977
REPORT       1#    191#
SCOPE       86#    549    1065    1301    1359    1420    1607    1707    1796    1896    2030    2084    2136    2271
SETPRI       1#      5#    191#   1338    1345    1398    1405
SETTRA    3944#   3953    3954    3955    3957    3959    3960    3961    3962    3963    3964
SETUP        1#    191#
SKIP         1#    191#
SLASH        1#    191#
SPACE      191#
STARS        1#      5#    191#    252     253     254     265     266     267     272     273     274     293     294     295
           386     397    2510    2511    2515    2516    2713    2731    3141    3273    3296    3305    3330    3338    3349
          3358    3367    3422    3484    3499    3519    3532    3579    3658    3735    3780    3785    3853    3882    3923
          3967    3983    4012    4030    4091    4348
SWRSU        1#    191#
TRMTRP    3944#
TYPBIN       1#    191#
TYPDEC       1#    191#
TYPNAM       1#    191#
TYPNUM       1#    191#
TYPOCS       1#      5#    191#    472    2462    2479    2844    3004    3199    3213    3222    3229    3256
TYPOCT       1#    191#   3803
TYPTXT       1#    191#
$$ESCA       1#    191#
$$NEWT       1#    191#
$$SET     3944#   3953    3954    3955    3957    3959    3960    3961    3962    3963    3964
$$SKIP       1#    191#
.EQUAT       1#      5#     81
.HEADE       1#      5#      8
.KT11        1#
.SETUP       1#      5#    241
.SWRHI       1#
.$ACT1       1#
.$APTB       1#
.$APTH       1#
.$APTY       1#
.$ASTA       1#
.$CATC       1#
.$CMTA       1#
.$DB2D       1#      5#   4028
.$DB20       1#
```

```
.$DIV       1#
.$EOP       1#
.$ERRO      1#
.$ERRT      1#
.$MULT      1#
.$POWE      1#      5#   3965
.$RAND      1#
.$RDDE      1#
.$RDOC      1#
.$READ      1#      5#   3778
.$R2AZ      1#
.$SAVE      1#      5#   3733
.$SB2D      1#      5#   4010
.$SB2O      1#
.$SCOP      1#
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#      5#   3921
.$TYPB      1#
.$TYPD      1#
.$TYPE      1#      5#   3577
.$TYPO      1#      5#   3656
.$40CA      1#
.1170       1#


. ABS.  017656     000


ERRORS DETECTED:  0

DSKZ:CZRXBF,DSKZ:CZRXBF.SEQ/CRF/SOL=[400,1066]SYSMAC.SML,[400,2465]CZRXBF.P11
RUN-TIME:  36 49 3 SECONDS
RUN-TIME RATIO:  363/90=4.0
CORE USED:  33K  (65 PAGES)
```